Christian-Albrechts-Universität zu Kiel

Diplomarbeit

Transformation von Esterel nach SyncChart

cand. inform. Lars Kühl
(Mat.Nr. 434142)

22. Februar 2006

Institut für Informatik und Praktische Mathematik
Lehrstuhl für Echtzeitsysteme und Eingebettete Systeme

Prof. Dr. Reinhard von Hanxleden

betreut durch:
Steffen H. Prochnow

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe.

Kiel,

_____

Zusammenfassung

Im Rahmen des Projektes *KIEL* wurde in diese Arbeit und zwei Module zum Projekt *KIEL* erstellt. Der erste Teil dieser Arbeit beschreibt die Transformation von Esterel Code in SyncCharts, wobei in dieser Arbeit der Transfomation eine größere Bedeutung zukommt. Dadurch, dass die Transformation der einzelnen Esterel-Befehle allgemein und einfach kombinierbar sind, wurde ein zweiter Teil, die Optimierung, von SyncCharts erforderlich. Im Folgenden wird die Transformation jedes Esterel-Befehls und die Optimierung von SyncCharts beschrieben.

Schlüsselwörter Esterel, EsterelStudio, cec

Inhaltsverzeichnis

Abbildungsverzeichnis

Verzeichnis der Auflistungen

Verzeichnis der Akronyme

KIEL Kiel Integrated Environment for Layout

CEC Columbia Esterel Compiler

EBNF Erweiterte Backus-Naur-Form

XML Extended Markup Language

UML Unifed Modelling Language

Verzeichnis der Transformationsregeln

Verzeichnis der Optimierungsregeln

1. Einleitung

Wir leben in einer Zeit, die durch die Verbesserungen in der Chiptechnologie geprägt ist. Diese Technologie gewinnt seit Jahren einen immer größeren Einfluss auf unser tägliches Leben. In einem Großteil der Haushalte der Industriestaaten steht mittlerweile ein Computer mit Internetanschluß, sodass Informationen weltweit zu beziehen sind, das Handy ermöglicht eine Erreichbarkeit an fast jedem beliebigen Ort, selbt Kaffeemaschinen produzieren heutzutage an Hand eines mit einem Barcode versehenen, mit Pulver gefülltem Plastikbehälters das gewünschte Heißgetränk, das weitverbreitete Navigationssystem weist den Weg zum dem gewünschten Ort und der Airbag verhindert oft schwerwiegende Folgen eines Unfalls. Diese zuletzt aufgezählten Systeme sind sogenannte *eingebettete Systeme*, die auf äußere Gegebenheiten reagieren. Fast jeder Gegenstand, der heutzutage genutzt wird und der auf irgendeine Art elektrisch betrieben wird, ist ein eingebettetes System. Überall auf diesen Systemen ist programmierbare Hardware vorhanden, auf der, je nach Anwendung, andere Programme laufen, um das System zu steuern. Diese Programme werden durch *Programmiersprachen* beschrieben. Der Duden [13] definiert eine Programmiersprache als

> ein System von Wörtern u. Symbolen, die zur Formulierung von Programmen dient.

Die ersten Programme wurden gegen Mitte des 19. Jahrhunderts auf *Lochstreifen* geschrieben.

Abbildung 1.1.: Lochstreifen Programm

Auf Bild 1 ist zu erkennen, dass diese Programme recht schwierig zu verstehen, erstellen oder zu verändern sind. Ebenso wenig können damit komplizierte Sachver-

halte in einer vernünftigen Länge beschrieben werden.

Der nächste große Schritt war dann die Beschreibung von Programmen durch Schalter, die auf An und Aus gestellt werden konnten. Dadurch wurde zwar eine größere Flexibilität erlangt, intuitiv verstanden und erstellt werden konnten diese Programme jedoch nach wie vor nicht.

Mitte des 20. Jahrhunderts wurden die Rechner leistungsfähiger und es wurden die Maschinensprachen, im englischen *Assembler*, entwickelt. Jene ermöglichten es nun, Programme in einer verständlicheren Form zu erstellen. Intern wurden die Befehle trozdem als eine Folge von Zuständen An und Aus gesehen.

Auflistung 1.1: Maschinensprache

```
01:read x
02:get  x
03:gotoz 10
04:subi 1
05:put  i
06:add  x
07:put  x
08:get  i
09:goto 03
10:write x
11:halt
```

Die Auflistung 1.1 ist ein Programm, welches die Summe der Zahlen von 1 bis zu einem eingelesenen Wert größer 0 berechnet. Die benutzten Befehle sind nur ein Beispiel dafür, wie ein Maschinenbefehl aussehen kann, da es verschiedene Maschinensprachen für die unterschiedlichen Chiparchitekturen gibt. Auf den Maschinensprachen aufbauend wurden die höheren Programmiersprachen wie z. B. : Fortran, COBOL, ALGOL, *C*, PASCAL und LISP entwickelt. Die Compiler für die jeweiligen Programmiersprachen wandeln dann diesen Programmtext in Programme der entsprechenden Maschinensprache so um, dass die Hardware nun genau das im Programmtext Beschriebene ausführt. In der Auflistung 1.2 ist das bereits dargestellt Beispiel 1.1 in der Sprache PASCAL aufgeführt.

Die Funktion des Programmes ist nun bereits erkennbar und Fehlersuche und Korrektur im Programmtext sind besser möglich. Es wurden noch viele dieser textlichen Programmiersprachen entwickelt, die jeweils auf spezielle Bedürfnisse in ihrer Anwendung eingehen.

Die Sprache Esterel [7, 6, 4, 5] ist eine von diesen, wobei Esterel gerade für die Beschreibung von eingebetteten Systemen geeignet ist. Die verschiedenen Esterel Compiler [8, 29, 15] übersetzen Esterel Programmcode in der Regel in *C*-Code, wobei vom Entwickler anschließend noch verschiedene Funktionalitäten implementiert werden müssen. Das *C*-Programm wird dann in den Maschinencode für die entsprechende Hardware übersetzt.

Auflistung 1.2: Summe in Pascal

```
program summe;
var
 i,n,x : integer;
begin
 read(n);
 x:= 0;
 for i:=1 to n do x= x+n;
 write(x);
end.
```

Dieses System des Übersetzens in eine andere Programmiersprache ist aufgrund der Kosten und Komplexität der Entwicklung einer kompletten, richtigen und effizienten Übersetzung von einer neuen Hochsprache in die verschiedenen Maschinensprachen, entstanden. Seit ein paar Jahren werden immer mehr Programme in grafischen Modellierungswerkzeugen wie EsterelStudio [16], Matlab [25] oder UML [33] erstellt. Im Unterschied zu textlichen Programmen, welche in der Regel aus Zeichen, die auf einer normalen Tastatur abgebildet sind, bestehen, nutzt ein Modellierungswerkzeug zusätzlich noch grafische Elemente wie Kreise, Rechtecke, Pfeile, Linien usw. zur Beschreibung eines Programmes. Mit Hilfe des Modellierungswerkzeugs wird eine Art „Zeichnung“ erstellt, welche ein Programm darstellt. Diese Zeichnungen werden nun wieder in eine Hochsprache übersetzt, in welcher teilweise noch Funktionalitäten beschrieben oder ergänzt werden müssen. Diese Zeichnungen oder im Fall dieser Arbeit ein SyncChart [1, 2], die nicht-kommezielle Variante der EsterelStudio -*Safe-State-Machine* [3] sind ein Dialekt der Zustands-Übergangs-Diagramme [20], im folgenden mit ihrem geläufigeren englischen Namen *Statechart* [19] bezeichnet. Sie sind in der Regel verständlicher als äquivalente textliche Programme. Ein Vergleich von einem Esterel-Programm und einem SyncChart ist in der Abblidung 1.2 dargestellt.

Das Verhalten der Programme kann sicherlich schneller an Hand des SyncCharts in Abbildung 1.2(b) erkannt werden, als in der textlichen Auflistung 1.2(a) in Esterel. Der Mensch versteht Bilder in der Regel intuitiver als einen Text und einen Text leichter als Zahlenkolonnen oder Formeln. Ebenso fällt es ihm leichter einen Fehler in einer Zeichnung zu finden als in einem Text oder einer Formel. In diesen Fall trifft das Sprichwort „ein Bild sagt mehr als tausend Worte “zu.

Ironischerweise gelingt es uns schneller eine Formel oder einen Text zu schreiben, als eine Zeichnung mit einer äquivalenten Aussage anzufertigen. Aus diesen Erkenntnissen entstand die Idee zu dieser Arbeit, in der die Vorteile einer graphischen Darstellung als SyncChart und der textlichen Erstellung als Esterel-Programm verbunden werden sollen.

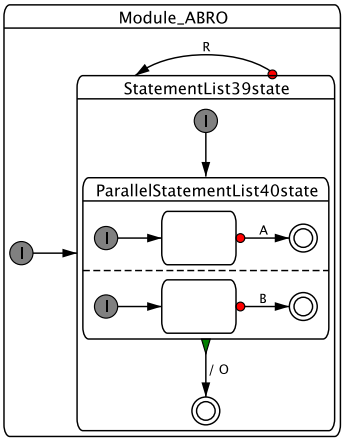
Die Arbeit wird folgendes umfassen.

- Kapitel 2 beschreibt die Grundlagen der Arbeit.

Auflistung 1.3: ABRO

```
module abro:
input A,B,R;
output O;
loop
 abort
   [ await A
     ||
     await B];
  emit O;
  halt
 when R
end loop
end module
```

(a) ABRO in Esterel



(b) ABRO als Statechart

Abbildung 1.2.: Die Gegenüberstellung von Esterel Programmcode und einem entsprechenden SyncChart

- Kapitel 3 beschreibt die Transformationen der einzelnen Esterel-Befehle in SyncCharts.
- Kapitel 4 beschreibt die Regeln und das Vorgehen der Vereinfachung der erzeugten SyncCharts.
- Kapitel 5 gibt ein Beispielhafte Umwandlung an.
- Kapitel 6 stellt die Implementierung und die verwendeten Hilfsmittel stelle vor.
- Kapitel 7 fasst die Arbeit zusammen und führt Möglichkeiten der Fortsetzung auf.
- Im Anhang B sind die Einstellungsmöglichkeiten des Programms beschrieben, im Abhang C ist der Quellcode aufgelistet.

2. Grundlagen

In dieser Arbeit wird die Umwandlung von einer textlichen Programmiersprache Esterel in graphische SyncCharts beschrieben. Die Programmiersprache Esterel ist, laut *Berry* [6], eine textliche Sprache zum Beschreiben von kontrollorientierten, reaktiven Systemen. Die SyncCharts sind eine graphische Beschreibung von kontrollorientierten, reaktiven Systemen. Ein solches System beschreibt die Reaktion auf äußere Einflüsse.

Die Umwandlung ist nur denkbar, da die Programmiersprache und das Modellierungswerkzeug dafür geeignet sind, dieselben Systeme zu beschreiben. Im Modellierungswerkzeug EsterelStudio ist die Umwandlung [28] aus SyncChart in Esterel bereits integriert. Die konträre Transformation soll hier erarbeitet werden.

2.1. Vorgaben

Die Vorgaben zur Transformation von Esterel in ein SyncChart bei dieser Arbeit waren:

- Erzeugen von SyncCharts aus Esterel,
- die Korrektheit der Umwandlung,
- gegebenenfalls die Vereinfachung des SyncChart,
- die Implementation der Umwandlung als Komponente von *KIEL*.

Bedingt durch diese Vorgaben war der erste Schritt für jeden Esterel-Kernel-Befehl eine Darstellung als SyncChart zu erarbeiten, um festzustellen, ob die Problemstellung lösbar ist, da noch keine Arbeit darüber verfasst worden ist. Die einzelnen Umwandlungen sind jeweils bei den Esterel -Befehlen 3 zu finden. Da sich die Kernel-Befehle in äquivalente SyncCharts transferieren ließen, wurden die Überlegungen auf die verbleibenden Befehle erweitert. Diese Befehle mussten ebenfalls umwandelbar sein, da jeder Nicht-Kernel-Befehl durch eine Kombination von Kernel-Befehlen ersetzbar ist.

Um die Korrektheit der Transformation zu zeigen, wurden nun die von EsterelStudio aus einem SyncChart erzeugten Esterel Module herangezogen. So war es nicht notwendig, diese Umwandlungen, welche auf der Basis von [28] aufgebaut sind, eigenhändig durchzuführen, sondern es war möglich auf die korrekte und bereits optimierte Darstellung eines Esterel-Programms zurückzugreifen. Um die Äquivalenz zwischen dem Esterel-Befehl und dem aus EsterelStudio heraus erzeugten Programm zu zeigen, wird im Folgenden die *Esterel-Process-Calculus-Syntax* aus [4].

Die Implementierung, die im Kapitel 6 genauer beschrieben wird, erfolgte im Rahmen des Projektes *KIEL* [32], welches mehrere Arbeiten am Lehrstuhl für Echtzeitsysteme und Eingebettete Systeme der Christian-Albrechts-Universität zu Kiel verbindet. Zunächst werden nun die Sprache Esterel und die graphischen Elemente eines SyncCharts vorgestellt.

2.2. Esterel

Die Sprache Esterel wurde von *Berry* [17, 7] zum Beschreiben von reaktiven Systemen entwickelt. Im Rahmen dieser Arbeit ist es nicht möglich detaillierte Informationen über Esterel zu geben, diese sind in *Berry* [5] nachzulesen.

Esterel führt Programmstücke in Zeiteinheiten, sogenannten Instanzen, aus. Das bedeutet, dass beliebig viele Befehle in einer Instanz ausgeführt werden, es sei denn, ein Befehl wartet auf den Beginn einer neuen Instanz. In diesen Fall werden die folgenden Befehle auch erst in der nächsten Instanz ausgeführt. Desweiteren reagiert ein Esterel Programm auf seine Umwelt, indem es eingehende Signale auswertet. Befehle, die auf diese Signale reagieren, beginnen mit der Überprüfung der Signale in der Regel in der, auf ihren Aufruf folgenden, Instanz. Um die Signale in der Instanz ihres Aufrufs auszuwerten, wird das Schlüsselwort *immediate* verwendet.

Nachfolgend wir eine Einführung in die Syntax von Esterel gegeben. Ein Esterel-Programm besteht aus einem Hauptmodul und eventuell aus weiteren Modulen. Der Name des Hauptmoduls ist auch gleichzeitig der Name des Programms. Anschließend folgt ein Deklarationsteil, der später im Abschnitt 2.2.1 näher beschrieben wirde. Als letztes folgt dann der Anweisungsteil, der aus mindestens einem Esterel-Befehl besteht. Die einzelnen Esterel-Befehle beschreibe werden im Zusammenhang mit der Transformation nach SyncCharts beschrieben, da dort das jeweilige Verhalten der Befehle umgesetzt wird.

Ist ein Esterel-Programm in mehrere Module unterteilt, so ist dies nur eine Vereinfachung für den Entwickler. Man kann aus einem Modul mittels des Befehls `run` ein anderes Modul aufrufen. Dieser `run`- Befehl wird dann textlich durch den Programmteil von dem aufgerufenen Modul ersetzt. Zudem werden übergebene Signalnamen im aufgerufenen Modul textlich durch die angegebenen, bereits vorhandenen Signalnamen ersetzt. Die Deklarationen des aufgerufenen Moduls werden zum Deklarationsteil des aufrufenden Moduls hinzugefügt. Der Befehl `run` ist somit mehr ein Hilfsmittel zur Gestaltung übersichtlicherer Programme. Da eine Textersetzung eher einem Parser [27] zuzuordnen ist und das Verhalten eines Programmes nicht beeinflußt, wird nachfolgend aus Gründen der Vereinfachung davon ausgegangen, dass ein Esterel-Programm nur aus einem Modul besteht. Dies ist jedoch der einzige Befehl, der nicht betrachtet wird, da die Textersetzung insbesondere in der Implementation durch den *cec* 6 gelöst wird.

Desweitern können *input, output, return, inputoutput* Signale, Funktionen, Typen, Prozeduren, Tasks und Relationen deklariert werden. Die *input, output, inputoutput* und *return* Signale definieren die möglichen globalen Signale mit Typ und Funktion,

wobei *Input*-Signale eingelesen werden, emphOutput-Signale ausgegeben werden, *InputOutput*-Signale eingelesen und ausgegeben werden und jedes *Return*-Signal beim Aufruf eines Tasks an diesen gebunden wird. Prozeduren und Funktionen führen ein Programm instantan aus und liefern im Falle von Funktionen ein Ergebnis zurück. Tasks varhalten ebenso müssen aber nicht instantan sein, sondern signalisieren ihre Termination durch ein bestimmtes *Return*-Signal.

Um die Sprache genau vorzustellen wird im Folgenden eine kontextfreie Grammatik [9, 10]die EBNF [24] verwendet.

2.2.1. Deklarationen

Definition 1 Der Deklarationsteil ⟨*DeclarationList*⟩ in einem Esterel Module ist folgendermaßen aufgebaut.

⟨*Ziffer*⟩ ::= ‘0’|‘1’|‘2’|‘3’|‘4’|‘5’|‘6’|‘7’|‘8’|‘9’

⟨*UnsignedIntegerConstant*⟩ ::= <Ziffer> {<Ziffer>}

⟨*Letter*⟩ ::= ‘a’ | ‘b’ |‘c’ |‘d’ |‘e’ |‘f’ |‘g’ |‘h’ |‘i’ |‘j’ |‘k’ |‘l’ |‘m’ |‘n’ | ‘o’ | ‘p’ |‘q’ |‘r’ |‘s’ |‘t’ |‘v’| ‘w’ | ‘x’ |‘y’ |‘z’ |‘_’ |‘A’| ‘B’ |‘C’ |‘D’ | ‘E’ |‘F’ |‘G’ |‘H’ |‘I’ |‘J’ |‘K’ |‘L’ |‘M’ |‘N’ |‘O’ | ‘P’ |‘Q’ |‘R’ | ‘S’ |‘T’ |‘V’ |‘W’ |‘X’ |‘Y’ |‘Z’

⟨*Identifier*⟩ ::= <Letter> {(<Letter>) | <Ziffer> }

⟨*TypeIdentifier*⟩ ::= <Identifier>

⟨*FunctionIdentifier*⟩ ::= <Identifier>

⟨*ProcedureIdentifier*⟩ ::= <Identifier>

⟨*TaskIdentifier*⟩ ::= <Identifier>

⟨*SignalIdentifier*⟩ ::= <Identifier>

⟨*VariableIdentifier*⟩ ::= <Identifier>

⟨*ConstantIdentifier*⟩ ::= <Identifier>

⟨*StringConstant*⟩ ::= <Identifier>

⟨*DeclarationList*⟩ ::= ‘type’ <TypeIdentifier>
{ ‘,’ <TypeIdentifier>}
| ‘constant’
<ConstantIdentifier> [‘=’<ConstantAtom>]
‘:’ <TypeIdentifier>
{‘,’<ConstantIdentifier> [‘=’<ConstantAtom>]
‘:’ <TypeIdentifier> }
| ‘function’ <FunctionIdentifier>
‘(’<TypeIdentifierList>‘)’ ‘:’<TypeIdentifier>
{‘,’ <FunctionIdentifier>
‘(’<TypeIdentifierList>‘)’ ‘:’<TypeIdentifier>}
| ‘procedure’ <ProcedureIdentifier>
‘(’<TypeIdentifierList>‘)
(’<TypeIdentifierList>‘)’
{‘,’ <ProcedureIdentifier>
‘(’<TypeIdentifierList>‘)
(’<TypeIdentifierList>‘)’}
| ‘task’ <TaskIdentifier>
‘(’<TypeIdentifierList>‘)
(’<TypeIdentifierList>‘)’
{‘,’ <TaskIdentifier>
‘(’<TypeIdentifierList>‘)
(’<TypeIdentifierList>‘)’}
| ‘input’ <SignalDeclList> ‘;’
| ‘inputoutput’ <SignalDeclList>‘;’
| ‘output’ <SignalDeclList>‘;’
| ‘return’ <SignalDeclList>‘;’
| ‘sensor’ <Identifier> ‘:’
<TypeIdentifier>
{‘,’<Identifier> ‘:’ <TypeIdentifier>}
| ‘relation’ (<Implication> | <Exclusion>)
{‘,’ (<Implication> | <Exclusion>)}

Die Nicht-Terminalzeichen ⟨*TypeIdentifierList*⟩, ⟨*SignalDeclList*⟩, ⟨*ChannelType*⟩, ⟨*Implication*⟩ und ⟨*Exclusion*⟩ sind folgendermaßen definiert.

⟨*TypeIdentifierList*⟩ ::= [<TypeIdentifier>]
| <TypeIdentifier> {‘,’<TypeIdentifier>}

⟨*SignalDeclList*⟩ ::= <SignalIdentifier> [‘:=’ <Expression>] [‘:’ <ChannelType>] {‘,’<SignalIdentifier> [‘:=’ <Expression>] [‘:’ <ChannelType>]}

⟨*ChannelType*⟩ ::= <TypeIdentifier>
| ‘combine’ <TypeIdentifier> ‘with’ (FunctionIdentifier | ‘+’ | ‘*’ | ‘and’ | ‘or’)

⟨*Implication*⟩ ::= <SignalIdentifier> ‘=>’ <SignalIdentifier>

⟨*Exclusion*⟩ ::= <SignalIdentifier> ‘#’ <SignalIdentifier> {‘#’ <SignalIdentifier>}

2.2.2. Ausdrücke

In diesem Abschnitt werden die verschiedenen Ausdrücke definiert.

Definition 2 Die Ausdrücke ⟨*Expression*⟩ sind folgendermaßen in der EBNF definiert:

⟨*Expression*⟩ ::= <Constant>
| ‘(’<Expression>‘)’
| ‘?’<SignalIdentifier>
| ‘pre(?’<SignalIdentifier>‘)’
| ‘??’<ExceptionIdentifier>
| ‘-’<Expression>
| ‘not’<Expression>
| <Expression> ‘*’ <Expression>
| <Expression> ‘/’ <Expression>
| <Expression> ‘+’ <Expression>
| <Expression> ‘-’ <Expression>
| <Expression> ‘mod’ <Expression>
| <Expression> ‘=’ <Expression>
| <Expression> ‘<>’ <Expression>
| <Expression> ‘<’ <Expression>
| <Expression> ‘<=’ <Expression>
| <Expression> ‘>’ <Expression>
| <Expression> ‘>=’ <Expression>
| <Expression> ‘and’ <Expression>
| <Expression> ‘or’ <Expression>
| <VariablenIdentfier>

⟨*Constant*⟩ ::= <ConstantLiteral>
| <UnsignedNummber>

⟨*ConstantLiteral*⟩ ::= <ConstantIdentifier>
| 'true'
| 'false'
| <StringConmstant>

⟨*ConstantAtom*⟩ ::= <ConstantLiteral>
| <SignedNumber>

⟨*SignedNumber*⟩ ::= ['-'] <UnsignedNumber>

⟨*UnsignedNumber*⟩ ::= <UnsignedIntegerConstant>
| <UnsignedFloatConstant>
| <UnsignedDoubleConstant>

⟨*UnsignedFloatConstant*⟩ := {<UnsignedIntegerConstant>}
(<UnsignedIntegerConstant> '.'
| '.' <UnsignedIntegerConstant>)
{<UnsignedIntegerConstant>} 'f'

⟨*UnsignedDoubleConstant*⟩ := {<UnsignedIntegerConstant>}
(<UnsignedIntegerConstant> '.'
| '.' <UnsignedIntegerConstant>)
{<UnsignedIntegerConstant>}

Als nächstes folgen die Darstellungen von den Signalausdrücken.

Definition 3 Die Ausdrücke für Signale, ⟨*SignalExpression*⟩, haben den folgenden Aufbau.

⟨*SignalExpression*⟩ ::= <SignalIdentifier>
| 'pre('<SignalIdentifier>')'
| 'not' <SignalExpression>
| <SignalExpression> 'and' <SignalExpression>
| <SignalExpression> 'or' <SignalExpression>
| '('<SignalExpression>')'

Definition 4 ⟨*DelayExpression*⟩ werden aus der folgenden EBNF aufgebaut.

⟨*DelayExpression*⟩ ::= [('immediate' | <Expression>)]
(<SignalIdentifier> | '[' <SignalExpression>']')

Nachdem nun die Deklarationen und Ausdrücke von Esterel bekannt sind, werden die Nicht-Terminale für die einzelnen Befehle vorgestellt, für welche die Ersetzungsregeln im Kapitel 3 beschrieben werden. Zum Abschluß des Abschnittes folgt die Definition von ⟨*Statement*⟩.

Definition 5

⟨*Statement*⟩ ::= <Sequence>
| <Parallel>
| <AtomicStatement>

⟨*AtomicStatement*⟩ ::= <Abort> | <Await> | <Assign>
| <DoWatching> | <DoUpTo>| <Halt>
| <Pause> | <Nothing> | <Emit>
| <Every> | <Exit> | <If>
| <LocalSignalDeclaration> |
| <LocalVariableDeclaration>
| <LoopEach> | <Loop>
| <Call> | <Exec> | <Present>
| <Repeat> | <Suspend> | <Sustain>
| <Trap> | '('<Statement>')'

Im folgenden Abschnitt werden die verwendeten graphischen Elemente der SyncCharts beschrieben.

2.3. Einführung in die Syntax der SyncChart

Dies soll keine vollständige Beschreibung der SyncCharts werden, sondern nur das Verständnis, der im späteren Verlauf abgebildeten SyncCharts, ermöglichen.

Die im Abschnitt 2.2.1 vorgestellten Deklarationen existieren ebenfalls in Esterel-Studio Statecharts.

Ein Statechart besteht aus Zuständen und Transitionen, wobei letztere die Zustände miteinander verbinden und eine Richtung besitzen. Ein SyncChart ist eine Form eines Zustands-Übergangs-Diagramms [20]. Im folgenden werden diese beiden Komponentenarten für ein SyncChart vorgestellt, wobei das Augenmerk zunächst auf die Zuständen gerichtet wird.

2.3.1. Zustände

In SyncChart gibt es Zustände und Pseudozustände. In einem Zustand kann das System beliebig lange verweilen, während die Pseudozustände immer sofort durchlaufen werden.

Desweiteren existieren *Makrozustände*, die wiederum ein Netz aus Zuständen und Transitionen beinhalten können. Die Makrozustände bilden so eine Hierachie innerhalb eines SyncChart. An dieser Stelle werden die Begriffe *normal termination*,*weak abortion* und *strong abortion* eingeführt, sie werden im Abschnitt 2.3.2 beschrieben.

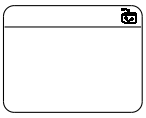

: Ein Initialer Zustand ist ein Pseudozustand, der den Startpunkt innerhalb des Makrozustandes darstellt.

: Ein *Conditional* Pseudozustand, der eine sofortige Weiterleitung gestattet.

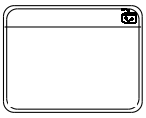
: Ein *Suspension* Pseudozustand, der mit ihm verbundene Zustände in der Ausführung einfrieren kann.

: Ein einfacher Zustand, den man nur mittels einer *strong abortion* oder *weak abortion* verlassen kann.

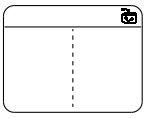
: Ein einfacher, finaler Zustand, der das Ende der Ausführung des übergeordneten Makrozustandes anzeigt.



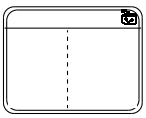
: Ein Makrozustand, der ein Statechart beinhaltet und so eine Hirarchie bildet. Von einem Makrozustand kann man mittels *strong abortion*, *weak abortion* und *normal termination* zu einem anderen Zustand gelangen.



: Ein finaler Makrozustand, der ein Statechart beinhaltet und nach dessen Ausführung, dem ihm übergeordneten Makrozustand anzeigt, dass dessen die Ausführung beendet ist.



: Ein paralleler Makrozustand, der mehrere parallele SyncChart beinhaltet. Haben diese jeweils einen finalen Zustand erreicht, so terminiert der parallele Zustand in der gleichen Instanz.



: Dieser finale, parallele Makrozustand verbindet die Eigenschaften des parallelen Makrozustandes sowie des finalen Makrozustandes indem nach seiner Terminierung auch der übergeordnete Makrozustand terminiert.

Neben diesen Zustandsarten sind noch Textblöcke und textliche Makrozustände in SyncCharts definiert, diese werden hier jedoch nicht verwendet. Die folgenden Transitionen verbinden die verschieden Zustände miteinander.

2.3.2. Transitionen

Neben den Zuständen existieren Verbindungen zwischen den Zuständen - die Transitionen. Eine Transition besitzt genau einen Anfangszustand und einen Endzustand, also eine Richtung. Jede Transition besitzt eine Bezeichnung, die beschreibt, ob und unter welchen Bediungen ein Zustand über diese Transition verlassen werden soll.

Eine solche Bezeichnung besteht aus mehreren Teilen und es ist abhängig von der Art der Transition, welche Teile vorhanden sind. Es folgt die Beschreibung einer vollständigen Transitionsbeschriftung.

⟨*TransitionLabel*⟩ ::= (‘#’ | ⟨*factor*⟩) ⟨*trigger*⟩ ‘{’⟨*condition*⟩‘}’ ‘/’ ⟨*effect*⟩

<factor> ist ein ganzzahliger Ausdruck, der dafür sorgt, dass eine Transition erst nach ⟨*factor*⟩-maligem Auftreten des ⟨*trigger*⟩ ausgelöst wird.

‘#’ steht für sofortige erstmalige Überprüfung des ⟨*triggers*⟩.

<trigger> ist ein Boolescher Ausdruck, der Signalausdrücke beinhalten kann.

<condition> ist ein Boolescher Ausdruck.

<effect> steht für beliebig viele instantane Esterel-Befehle.

Nachdem nun der Aufbau der Transitionsbeschriftungenbeschrieben ist, wird nun auf die verschiedenen Transitionen eingegangen.

Transitionsarten

Bevor nun zur Beschreibung der unterschiedlichen Transitionen übergegangen wird, muss darauf hingewiesen werden, dass jede Transition an jedem Zustand, mit Ausnahme der Pseudozustände *Initial* und *Suspension* enden kann.

Die Transitionen sind.

 : Die *Conditional*-Transition ist optisch nur durch den Startzustand von der *weak abortion*s und der Initialen Transitionzu unterscheiden, wobei die Initialen Transitionen in der Funktionalität der *Conditional*-Transitionen gleichen. Die *conditional*-Transition beginnt bei einem *Conditional*-Zustand. Es können beliebig viele *conditional*-Transitionen von einem Zustand ausgehen, wobei aber immer eine so genannte *default*-Transitionnotwendig ist, d.h. sie besitzt in der Transitionsbeschriftung weder einen <trigger>, noch eine <condition>.

 : Eine *Suspension* startet nur bei einem *Suspension*-Pseudozustand und führt zu dem Zustand, der eingefroren werden soll.

 : Eine *normal termination* kann nur an einem Makrozustand beginnen und es darf nur genau eine *normal termination* von einem Makrozustand ausgehen. Die Transitionsbeschriftung darf nur aus einem <effect> bestehen.

 : Eine *strong abortion* kann an jedem Nicht-Pseudozustand beginnen und bricht die Ausführung eines Zustandes sofort ab, sobald der <trigger> erfüllt ist.

 : Eine *weak abortion* kann an jedem Nicht-Pseudozustand beginnen und bricht die Ausführung eines Zustandes am Ende einer Instanz ab, sobald der <trigger> erfüllt ist.

Eine weitere wichtige Eigenschaft von Transitionen bilden die im Folgenden Abschnitt beschriebenen Prioritäten.

Prioritäten

Zusätzlich zu den Transitionsbeschriftungen hat jede Transition eine *Priorität*, dargestellt durch eine natürliche Zahl. Eine Transition hat eine hohe Priorität, wenn der Wert möglichst niedrig ist, wobei der Wert einer Priorität immer mindestens 1 ist und kein Wert, ausgehend von ein und dem selben Zustand, mehrfach vergeben werden darf. Eine weitere Bedingung bei den Prioritäten von Transitionen, welche an einem Zustand beginnen, ist, dass eine *strong abortion* eine höhere Priorität hat

als eine *weak abortion* und diese wiederum eine höhere als die *normal termination*.

Diese kurze Einführung in SyncCharts sollte für ein ausreichendes Verständnis der noch folgenden Teile dieser Arbeit sorgen.

2.4. Verwendete Programme

Bevor der Haupteil dieser Arbeit beginnt, soll noch ein verwendetes Programm und das bereits erwähnte Projekt *KIEL* vorgestellt werden.

2.4.1. Der *Columbia Esterel Compiler*

Der *Columbia Esterel Compiler* [14], kurz cec, ist ein freier Esterel Compiler. Er erstellt aus einem Esterel-Programm ein *C*-Programm, in welchem dann die Funktionalitäten von Signalen, Tasks, Prozeduren, Funktionen und Typen in *C* beschrieben werden müssen.

Der cec bietet zudem die Möglichkeit Zwischenschritte der Übersetzung von Esterel zu *C* als Datei abzuspeichern. Von besonderem Interesse ist dabei die `.exp` Datei, welche das expandierte Esterel-Programm, also ein Modul, indem sämtliche Aufrufe von Untermodulen mittels des `run`-Befehls, korrekt ersetzt wurden, als XML-Dokument enthält. Dadurch entfällt eine Erkennung von Syntaxfehlern, Typefehlern und unzulässige Zyklen, da der cec eine entsprechende Fehlermeldung generiert. Nachteilig ist jedoch, dass der cec nur Esterel Programme des Typs „Esterel v5“ erkennt und nicht „Esterel v7“.

Sollte der cec jedoch erweitert werden, so sollte dies nur Veränderungen beim Parsen der `.exp` Datei und einige Erweiterungen in der Transformation nachsich ziehen.

2.4.2. *KIEL*

KIEL steht für Kiel Integrated Environment for Layout und ist eine Entwicklung des Lehrstuhls für Echtzeitsysteme und Eigebette te Systeme der Christian-Albrechts-Universität zu Kiel [11]. Es soll die Darstellung von Statecharts, die in unterschiedlichen Werkzeugen entstanden sind, vereinheitlichen, um so eine bessere Lesbarkeit der Statecharts zu erreichen.

Die Grafik 2.4.2 bietet eine Übersicht der Struktur von *KIEL*.

Da *KIEL* in *Java* [30] implementiert ist, ist es plattformunabhängig und erzwingt eine Paket-und Klassenstruktur für das Projekt. Die zu *KIEL* gehörenden *Java*-Pakete sind nach den in der Abbildung 2.4.2 dargestellten Modulen benannt. Durch diese Aufteilung in unterschiedliche Module ist eine parallele Enwicklung von *KIEL* möglich. Im Folgenden wird kurz auf die einzelnen Module eingegangen.

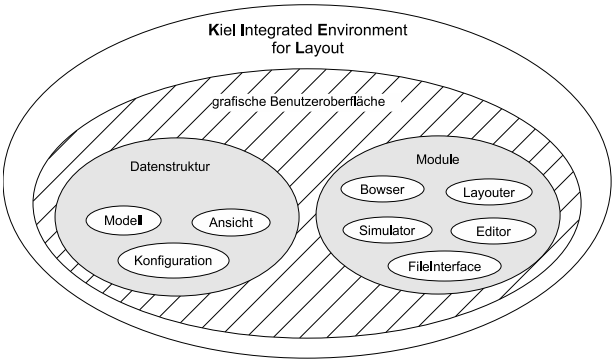


Abbildung 2.1.: Die Struktur von *KIEL*

Die *KIEL*-Datenstruktur

Die *KIEL*-Datenstruktur [31] ist das Zentrum Projektes *KIEL* und wurde zum Beginn entwickelt. Zu der kompletten Datenstruktur des Projektes *KIEL* gehören neben der eigentlichen Datenstruktur noch grafische Informationen und Informationen über den aktuellen Zustand eines Statecharts. Die beiden letzteren Pakete dienen der Visualisierung eines Statecharts, während in der eigentlichen Datenstruktur alle zulässigen Komponenten eines Statecharts beschrieben werden. Die *KIEL*-Datenstruktur vereint dabei mehrere Dialekte von Statecharts als Obermenge.

Der Editor

Der Editor ermöglicht es *Stateflow* [18] und SyncChart innerhalb von *KIEL* zu erstellen. Er gleicht in seinen Grundfunktionen dem Editor, der in EsterelStudio vorhanden ist, bietet darüber allerdings eine Vielzahl von Möglichkeiten, die man in der Arbeit [23] nachlesen kann.

Der Browser

Der Browser stellt ein Statechart dar. Er beinhaltet zusätzlich eine Schnittstelle zum Simulator, um die Simulation eines Statecharts unmittelbar grafisch zu betrachten. Desweiteren ermöglicht der Browser auch die Manipulation des Statecharts durch den im folgenden Abschnitt beschriebenen Layouter und den Optimierer 4. Diese Funktionen stehen auch im Editor zur Verfügung, dienen aber primär der Manipulation von syntaktisch korrekten Statecharts. Weitere Informationen zum Browser sind in der Arbeit [34] zu finden.

Der Layouter

Der Layouter wurde in den Arbeiten [21, 22] entwickelt und wird dort auch ausführlich erklärt. Der Layouter erstellt für die möglichen Zustände eines Statecharts

ein möglichst gut lesbares und einheitliches Layout. So werden zwei Statecharts, die die gleichen Zustände und Transitionen, aber unterschiedliche grafische Ausprägungen besitzen auf ein und dieselbe Art dargestellt. Dies erleichtert das Verständnis eines Statecharts. Der Layouter bietet mehrere Layoutvarianten an, die auch in der oben genannten Arbeiten beschrieben sind. Für diese Arbeit ist der Layouter von herausragender Bedeutung, da ein Esterel-Programm keine grafischen Informationen besitzt und der Layouter für Statecharts ohne grafisches Layout ein Layout und somit grafische Informationen erstellt.

Der Simulator

Der Simulator ermöglicht die Simulation eines Statecharts. Diese Simulation wird durch den Browser visuell angezeigt, indem die aktuellen aktiven Zustände und Transitionen farblich hervorgehoben und animiert werden. Je nach Statechart-Dialekt wird ein Simulator benötigt. Momentan existiert der Simulator, der aus der Arbeit [26] hervorgegangen ist.

Das Fileinterface

Das Fileinterface [34] dient zum Einlesen der verschieden Statechart-Dialekte in *KIEL*. Die Umwandung von Esterel in SyncCharts ist hier ebenfalls eingebunden. Es ist, neben der Datenstruktur, das wichtigste Element von *KIEL*, da hier die Umwandlung von externen Datenquellen in die *KIEL* -Datenstruktur stattfindet.

3. Transformation von Esterel in SyncCharts

3.1. Einführungen

Bevor die einzelenen Umwandlungen erläutert werden, müssen ein paar Erklärungen zu den Schreibweisen und der Aufteilung vorgenommen werden, sodass das Verständnis vereinfacht wird.

Da in den Grafiken und in den Beweisen, mehrere *Statement*s auftreten können, werden die folgenden alternativen Schreibweisen benutzt.

p	:	⟨*Statement*⟩
q	:	⟨*Statement*⟩
DE	:	⟨*DelayExpression*⟩
SE	:	⟨*SignalExpression*⟩
ES	:	⟨*ExceptionSignal*⟩
EXP	:	⟨*Expression*⟩
ID	:	⟨*VariableIdentifier*⟩
S	:	⟨*SignalIdentifier*⟩
PID	:	⟨*ProcedureIdentifier*⟩
VList	:	{⟨*VariableIdentifier*⟩}
EXPList	:	{⟨*Expression*⟩}
ExceptionSignalList	:	⟨*ExceptionDeclaration*⟩ {⟨*ExceptionDeclaration*⟩}

Tritt eine Indizierung hinter den Bezeichnern auf, so wird zwischen endlich vielen Vorkommen unterschieden.

3.2. Die Transformation der Esterel-Deklarationen

Die Deklarationen eines Esterel-Moduls werden für das SyncChart übernommen.

3.3. Die Transformation der Esterel-Ausdrücke

Da es in SyncCharts keine Ausnahmesignale gibt, müssen diese durch normale Signale ersetzt werden, wodurch der Ausdruck '`??`' in den Statecharts nicht vorhanden ist. Anstatt, wie in Esterel '`immediate`' wird in SyncCharts '`#`' verwendet. Demzufolge ist eine entsprechende textliche Ersetzung durchzuführen.

3.4. Die Transformation der Esterel-Befehle

Es wird bei jedem Esterel-Befehl nach dem gleichen Muster vorgegangen.

- Esterel-Befehl in der EBNF,
- die Beschreibung des Befehls,
- eine Beschreibung eines äquivalenten Makrozustandes,
- Aufführung einer Regel für die unten eingeführte Grammatik, links ein nichtterminaler textlicher Makrozustand mit einem Esterel Befehl,rechts ein grafischer Makrozustand mit derselben semantischen Funktion,
- Behauptung und Beweis, dass der Esterel Programmcode, welcher aus dem Makrozustand erzeugt wird, und der Esterel Befehl semantisch äquivalent sind.

Bei den Beweisen zur semantischen Äquivalenz von Programmstücken, werden mehrere Axiome und Folgerung aus [6] benutzt. Diese werden ebenso wie ihre abkürzende Schreibweise für Esterel-Befehle vorgestellt.

Behavior Calculus

Im diesen Abschnitt wird das *Behavior-Calculus* von Esterel vorgestellt und erweitert. Dazu werden zunächst die bekannten Schreibweisen und Regeln aufgeführt. Anschließend folgen Schreibweisen und Regeln für weitere Befehle.

`nothing`	0		
`pause`	1		
`emit S(exp)`	!S(exp)		
`present SE then p else q`	SE?p,q		
`p;q`	p;q		
`loop p end`	p^*		
`p || q`	p | q		
`signal S in p end`	p\S		
`suspend p when SE end`	SE⊃p		
`trap T in p end`	{p}		
`exit T`	k with $k \geq 2$		
	↑p		
`suspend p when immediate SE end`	SE· ⊃p	≡	{(SE?1,2)*};SE⊃p
`await immediate SE do p end`	SE· ⇒p	≡	{(SE(↑p;2),1)* }
`await SE do p end`	SE⇒p	≡	1;SE· ⇒p
`weakly abort p when immediate`	SE· >p	≡	{(↑p;2)| s· ⇒}
`weakly abort p when`	SE>p	≡	{(↑p;2)| s⇒}
`abort p when immediate`	SE· ≫p	≡	SE· >SE(SE· ⊃p)
`abort p when SE`	SE≫p	≡	SE>SE(SE⊃p)

Die folgenden Schreibweisen dienen zur Ergänzung, sei dazu $x \in \aleph$.

Definition 6

`present case` SE_1 `do` p_1
`...case` SE_n `do` p_n $\quad \mathsf{SE}_1,\ldots,\mathsf{SE}_n?\mathsf{p}_1,\ldots,\mathsf{p}_n$
`else q end`

Definition 7

`await` x SE `do p` $\quad x$ SE `-->` p Dazu gelte die Ersetzung

$$x\ \mathsf{SE}\ \texttt{-->}\ \mathsf{p} \equiv \begin{cases} \mathsf{SE} \Rightarrow (x-1)\ \mathsf{SE}\ \texttt{-->}\ \mathsf{p} & \text{, falls } x > 1 \\ \mathsf{SE} \Rightarrow \mathsf{p} & \text{, sonst} \end{cases}.$$

Definition 8

`await` DE `do p` $\quad \mathsf{DE}\rightsquigarrow\mathsf{p}$ Es gelte die Ersetzung

$$\mathsf{DE}\rightsquigarrow\mathsf{p} \equiv \begin{cases} x\mathsf{SE}\ \texttt{-->}\ \mathsf{p} & \text{, falls DE} = \text{<factor>}\ \mathsf{SE} \\ \mathsf{SE} \cdot \Rightarrow \mathsf{p} & \text{, falls DE} = \texttt{immediate SE} \\ \mathsf{SE} \Rightarrow \mathsf{p} & \text{, sonst} \end{cases}.$$

Definition 9

`await case` DE_1 `do` p_1
`...case` DE_n `do` p_n $\quad \mathsf{DE}_1,\ldots,\mathsf{DE}_n \not\Rightarrow \mathsf{p}_1,\ldots,\mathsf{p}_n$
`end`

Definition 10

`weak abort case` DE_1
`do` p_1
`...case` DE_n `do` p_n
`end`

$$(\{(\uparrow \mathrm{p};2) | \mathsf{DE}_1,\ldots,\mathsf{DE}_n \not\Rightarrow !(\mathrm{H}_1);2,\ldots(\mathrm{H}_n);2\};$$

$$(\mathrm{H}_1),\ldots,(\mathrm{H}_n)?\mathsf{p}_1,\ldots,\mathsf{p}_n)\setminus(\mathrm{H}_1)\ldots\setminus(\mathrm{H}_n)$$

Es folgen nun die Regeln für das Verhalten eines Befehls. Dazu sei E die Menge der Signale.

$$\text{(null)}\ \ 0 \xrightarrow[E]{\emptyset,0} 0$$

$$\text{(unit delay)}\ \ 1 \xrightarrow[E]{\emptyset,1} 0$$

$$\text{(emit)}\ \ !\mathsf{S} \xrightarrow[E]{\{S\},0} 0$$

Es folgen die Ableitungsregeln, beginnend mit der allgemeinen Form einer Ableitung, $\texttt{p} \xrightarrow[E]{E_p,k} \texttt{p'}$, wobei E_p die Menge der gesendeten Signale in `p` und k der *Completion-Code* von `p` ist.

$$\text{(seq1)}\ \frac{\mathtt{p} \xrightarrow[E]{E_p,k} \mathtt{p'}\ k \neq 0}{\mathtt{p;q} \xrightarrow[E]{E_p,k} \mathtt{p';q}}$$

$$\text{(seq2)}\ \frac{\mathtt{p} \xrightarrow[E]{E_p,0} \mathtt{p'} \qquad \mathtt{q} \xrightarrow[E]{E_q,k} \mathtt{q'}}{\mathtt{p;q} \xrightarrow[E]{E_p \cup E_q,k} \mathtt{q'}}$$

$$\text{(loop)}\ \frac{\mathtt{p} \xrightarrow[E]{E_p,k} \mathtt{p'}\ k \neq 0}{\mathtt{p}^* \xrightarrow[E]{E_p,k} \mathtt{p';p}^*}$$

$$\text{(parallel)}\ \frac{\mathtt{p} \xrightarrow[E]{E_p,k} \mathtt{p'} \qquad \mathtt{q} \xrightarrow[E]{E_q,l} \mathtt{q'}}{\mathtt{p|q} \xrightarrow[E]{E_p \cup E_q,max(k,l)} \mathtt{p'|q'}}$$

$$\text{(present+)}\ \frac{\mathtt{SE}^+ \in E\ \mathtt{p} \xrightarrow[E]{E_p,k} \mathtt{p'}}{\mathtt{SE?p,q} \xrightarrow[E]{E_p,k} \mathtt{p'}}$$

$$\text{(present-)}\ \frac{\mathtt{SE}^- \in E\ \mathtt{q} \xrightarrow[E]{E_q,k} \mathtt{q'}}{\mathtt{SE?p,q} \xrightarrow[E]{E_p,k} \mathtt{q'}}$$

$$\text{(sig+)}\ \frac{\mathtt{p} \xrightarrow[E*\mathtt{S}^+]{E_p*\mathtt{S}^+,k} \mathtt{p'}\ \mathtt{S}(E_p) = \mathtt{S}(E) \backslash S}{\mathtt{p} \backslash \mathtt{S} \xrightarrow[E]{E_p,k} \mathtt{p'} \backslash \mathtt{S}}$$

$$\text{(sig-)}\ \frac{\mathtt{p} \xrightarrow[E*\mathtt{S}^-]{E_p*\mathtt{S}^-,k} \mathtt{p'}\ \mathtt{S}(E_p) = \mathtt{S}(E) \backslash S}{\mathtt{p} \backslash \mathtt{S} \xrightarrow[E]{E_p,k} \mathtt{p'} \backslash \mathtt{S}}$$

$$\text{(exit)}\quad k \xrightarrow[E]{\emptyset,k} \mathtt{0}$$

$$\text{(trap1)}\ \frac{\mathtt{p} \xrightarrow[E]{E_p,k} \mathtt{p'}\ k = 0 \vee k = 2}{\{\mathtt{p}\} \xrightarrow[E]{E_p,0} \mathtt{0}}$$

$$\text{(trap2)}\ \frac{\mathtt{p} \xrightarrow[E]{E_p,k} \mathtt{p'}\ k = 1 \vee k > 2}{\{\mathtt{p}\} \xrightarrow[E]{E_p,\downarrow k} \{\mathtt{p'}\}}$$

$$\text{(shift)}\ \frac{\texttt{p} \xrightarrow[E]{E_p,k} \texttt{p'}}{\uparrow\texttt{p}\} \xrightarrow[E]{E_p,\uparrow k} \uparrow\texttt{p'}}$$

$$\text{(suspend1)}\ \frac{\texttt{p} \xrightarrow[E]{E_p,0} \texttt{p'}}{\texttt{SE}\supset\texttt{p} \xrightarrow[E]{E_p,0} \texttt{p'}}$$

$$\text{(suspend2)}\ \frac{\texttt{p} \xrightarrow[E]{E_p,k} \texttt{p'}\ k \neq 0}{\texttt{SE}\supset\texttt{p} \xrightarrow[E]{E_p,k} \texttt{SE}\cdot\ \supset\texttt{p'}}$$

Für den *completion code* k gilt

$$\downarrow k = \begin{cases} 0 & \text{, falls } k = 0 \vee k = 2 \\ 1 & \text{, falls } k = 1 \\ k-1 & \text{, falls } k > 2 \end{cases}$$

und

$$\uparrow k = \begin{cases} k & \text{, falls } k = 0 \vee k = 1 \\ k+1 & \text{, falls } k > 1 \end{cases} .$$

Für die Beweise zur semantischen Äquivalenz der Transformationssregeln werden die folgenden Lemmata als Hilfen.

Im Lemma 1 wird gezeigt dass ein `nothing` -Befehl keinen Einfluss auf die Semantik des Programmes hat, gleichgültig ob `nothing` vor oder nach einem Befehl steht.

Lemma 1 (nothing) *Sei* `p` *ein beliebiger Esterel Befehl und E die Menge der aktiven Signale, dann gilt:*

1. `nothing;p` $\Leftrightarrow$ `p`.
2. `p;nothing` $\Leftrightarrow$ `p`.

Beweis
Zu 1:
Es gilt:

$$\frac{\text{(null)}\ 0 \xrightarrow[E]{\emptyset,0} 0 \quad p \xrightarrow[E]{E_p,k} p'}{\text{(seq2)}\ 0;p \xrightarrow[E]{E_p,k} p'}$$

Da auch $p \xrightarrow[E]{E_p,k} p'$ gilt, ist $0;p \Leftrightarrow p$.
Zu 2:
Hierfür wird nur die Inztanz betrachtet, in welcher `p` terminiert, also $k = 0$ ist, da sich `p` vorher in allen Instanzen auf beiden Seiten gleich verhält. k kann nur durch die Regeln *(null)* oder *(emit)* auf 0 gesetzt werden, deshalb entspricht in diesen Fällen p' 0. Es gilt:

$$\frac{p \xrightarrow[E]{E_p,0} 0 \quad (\text{null})\ 0 \xrightarrow[E]{\emptyset,0} 0}{(\text{seq2})\ p;0 \xrightarrow[E]{E_p,0} 0}$$

Da $p \xrightarrow[E]{E_p,0} 0$ gilt, ist $p;0 \Leftrightarrow p$.

Definition 11 (presentcase) Die zu Definition 6 gehörenden Ableitungsregeln lauten:

$$\frac{i \in \{1,\ldots,n\}\ SE_i \in E \wedge \forall j < i : SE_j \notin E \quad p_i \xrightarrow[E]{E_{p_i},k} p_i'}{\mathsf{SE}_1,\ldots,\mathsf{SE}_n?\mathsf{p}_1,\ldots,\mathsf{p}_n \xrightarrow[E]{E_{p_i},k} p_i'}$$

und

$$\frac{i \in \{1,\ldots,n\}\ \forall i \leq n : SE_i \notin E \qquad q \xrightarrow[E]{E_q,k} q'}{\mathsf{SE}_1,\ldots,\mathsf{SE}_n?\mathsf{p}_1,\ldots,\mathsf{p}_n \xrightarrow[E]{E_q,k} q'}$$

Lemma 2 (presentCase)
Es gilt :
$\mathsf{SE}_1,\ldots,\mathsf{SE}_n?\mathsf{p}_1,\ldots,\mathsf{p}_n \Longleftrightarrow$
$\mathsf{SE}_1?\mathsf{p}_1,\mathsf{SE}_2?\mathsf{p}_2,\ldots,\mathsf{SE}_n?\mathsf{p}_n,\mathsf{q}$

Beweis
Sei $i \in \{1,\ldots,n\}$. Für den Beweis zur semantischen Äquivalenz werden zwei Fälle betrachtet. Zunächst der Fall, in dem keine der *Signalexpression* SE_i zutrifft und im Folgenden der Fall in dem die *Signalexpression* SE_i zutrifft und keine mit kleinerem Index.

Für den Fall $\mathsf{SE}_i \notin E$ gilt:

$$\frac{(\text{presentcase}) \quad i \in \{1,\ldots,n\}\ \forall i \leq n : SE_i \notin E\ q \xrightarrow[E]{E_q,k} q'}{\mathsf{SE}_1,\ldots,\mathsf{SE}_n?\mathsf{p}_1,\ldots,\mathsf{p}_n \xrightarrow[E]{E_q,k} q'}$$

ist äquivalent zu

$$\cfrac{\cfrac{\cfrac{\cfrac{(\text{present})\mathsf{SE}_n \notin E \qquad \mathrm{q} \xrightarrow[E]{E_q,k} q'}{(\text{present})\mathsf{SE}_{n-1} \notin E \qquad \mathsf{SE}_n?\mathsf{p}_n,\mathsf{q} \xrightarrow[E]{E_q,k} q'}}{\vdots \qquad\qquad \vdots}}{(\text{present})\ \mathsf{SE}_1 \notin E \qquad \mathsf{SE}_2?\mathsf{p}_2,\ldots,\mathsf{SE}_n?\mathsf{p}_n,\mathsf{q} \xrightarrow[E]{E_q,k} q'}}{\mathsf{SE}_1?\mathsf{p}_1,\mathsf{SE}_2?\mathsf{p}_2,\ldots,\mathsf{SE}_n?\mathsf{p}_n,\mathsf{q} \xrightarrow[E]{E_q,k} q'}$$

Für den Fall $\mathsf{SE}_i \in E$ und $\mathsf{SE}_j \notin E$ für alle $j < i$ gilt:

$$\frac{\text{(presentcase)} \quad i \in \{1,\ldots,n\}\ SE_i \in E \wedge \forall j < i : SE_j \notin E\ p_i \xrightarrow[E]{E_{p_i},k} p_i'}{\mathsf{SE}_1,\ldots,\mathsf{SE}_n?\mathsf{p}_1,\ldots,\mathsf{p}_n \xrightarrow[E]{E_{p_i},k} p_i'}$$

ist äquivalent zu

$$\frac{\begin{array}{ll}\text{(present)}\mathsf{SE}_i \in E & p_i \xrightarrow[E]{E_{p_i},k} p_i' \\ \text{(present)}\mathsf{SE}_{i-1} \in E & \mathsf{SE}_i?\mathsf{p}_i,\mathsf{q} \xrightarrow[E]{E_{p_i},k} p_i' \\ \vdots & \vdots \\ \text{(present) } \mathsf{SE}_j \notin E & \mathsf{SE}_{j+1}?\mathsf{p}_{j+1},\ldots,\mathsf{SE}_n?\mathsf{p}_n,\mathsf{q} \xrightarrow[E]{E_{p_i},k} p_i' \\ \vdots & \vdots \end{array}}{\mathsf{SE}_1?\mathsf{p}_1,\mathsf{SE}_2?\mathsf{p}_2,\ldots,\mathsf{SE}_n?\mathsf{p}_n,\mathsf{q} \xrightarrow[E]{E_{p_i},k} p_i'}$$

Definition 12 (awaitcase) Die Verhaltensregel zu $\mathsf{DE}_1,\ldots,\mathsf{DE}_n \not\Rightarrow \mathsf{p}_1,\ldots,\mathsf{p}_n$ in der bereits etablierten Schreibweise lautet:

$\mathsf{DE}_1,\ldots,\mathsf{DE}_n \not\Rightarrow \mathsf{p}_1,\ldots,\mathsf{p}_n \equiv (\{\mathsf{DE}_1 \rightsquigarrow !\mathsf{S}_1;2|\ldots|\mathsf{DE}_n \rightsquigarrow !\mathsf{S}_n;2\ \};\ \mathsf{S}_1,\ \ldots,\mathsf{S}_n?\mathsf{p}_1,\ \ldots,\mathsf{p}_n,0)\ )\backslash\ \mathsf{S}_1\ \ldots\ \backslash\ \mathsf{S}_n$

Trifft mindestens eine *Delayexpression* zu, dann wird zuerst ein Signal, welches den entsprechenden Fall repräsentiert, gesendet. Anschließend wird die *Trap* ausgelöst, also werden alle parallelen Ausführungen von `await` beendet. Da nun mehrere Signale aktiv sein können, wird durch *presentcase* aus Lemma 6 der Fall mit der höchsten Priorität gewählt und das entsprechende Statement ausgewählt. Der Fall, dass kein Signal aktiv ist, kann nicht eintreten da ein Signal S_i aktiv sein muss, wird aber durch ein `nothing` dargestellt.

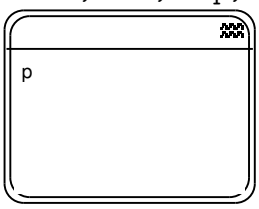
3.4.1. Die Transformationsgrammatik

Sollte bei einem Esterel-Befehl in der beschreibenden Grammatik ein optionales ⟨*Statement*⟩ vorhanden sein, und an einer Stelle im Code von diesen optionalen ⟨*Statement*⟩ kein Gebrauch gemacht werden, so wird ein `nothing` als Ersatz verwendet. Durch diese Vereinfachung kann die Anzahl der Regeln deutlich reduziert werden, denn das Nichtverwenden optionaler Statements dient lediglich der Vereinfachung des zu schreibenden Programms. Lemma 1 zeigt, dass zusätzliche `nothing`-Befehle keinen Einfluß auf das Verhalten haben.

Definition 13 Da bei den Deklarationen und den Ausdrücken keine Umwandlung von Text zu Grafik zu vollziehen ist, sondern die Textkomponenten kopiert werden, werden diese Teile der Sprache Esterel bei der fo lgenden Grammatik nicht verwendet. G sei die Transformationsgrammatik, V die Menge der Nichtterminal-Zeichen, `module` das Startsymbol und P die Menge der Transformationsregeln.

Sei $G = (V, \Sigma, \texttt{module}, P)$ mit

V = {`module, abort, assign, await,do upto, do watching, ||, halt,pause, nothing, emit,every,exit, if, signal,var, loopeach,loop, ||, present, repeat, positive, ;,suspend, sustain, call,exec,trap, weak`} $\cup$



Σ = Die unter Punkt 2.3 vorgestellten Grafiken für Zustände und Transitionen.

Die einzelnen Elemente der Transformationsregeln P werden in den folgenden Abschnitten beim jeweils zugehörigen Befehl angegeben.

3.4.2. Das Modul

Esterelgrammatik

⟨*Module*⟩ ::= '`module`' ⟨*Identifier*⟩ '`:`' {⟨*DelarationList*⟩} ⟨*Statement*⟩ '`end`' ['`module`']

Beschreibung des Befehls

Ein `module` beschreibt den Namen eines Programms, die Namen externer Ein- und Ausgaben und die Syntax möglicher extern zu definierender Typen, Funktionen, Tasks und Prozeduren. Zudem werden Relationen zwischen den Signalen definiert. Die verschieden Deklarationen sind im Abschnitt 2.2.1 erläutert.

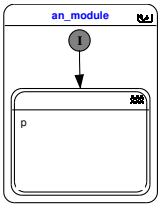
Beschreibung eines äquivalenten Makrozustandes

Die Deklarationen werden vom SyncChart übernommen. Zudem wird der Makrozustand, welcher aus der Transformation des ⟨*Statement*⟩ entsteht, von der Initialen Transition referenziert.

Transformationsregel 1 (module)

module <Identifier>
<DeclarationList>
p
end module

$\longmapsto$



Lemma 3 *Es gilt die folgende Äquivalenz.*

```
module <Identifier>:
<DeclarationList>
p
end module
```

$\Longleftrightarrow$

```
module <Identifier>:
<DeclarationList>
 nothing;
 [p];
halt
end module
```

BEWEIS Entfernt man den `nothing`-Befehl aufgrund des Lemma 1, so ergibt sich, dass auf beiden Seiten dieselben Programmstücke stehen. Diese sind auch semantisch äquivalent.

3.4.3. Der Befehl nothing

Esterelgrammatik

⟨*Nothing*⟩ ::= 'nothing'

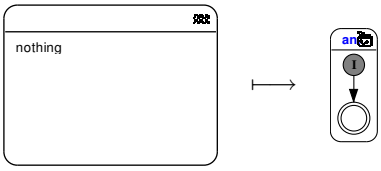
Beschreibung des Befehls

Der `nothing` Befehl terminiert sofort.

Beschreibung eines äquivalenten Makrozustandes

Eine sofortige Terminierung eines Makrozustandes wird durch einen direkten Übergang vom initialen Zustand zu einem einfachen, finalen Zustand erreicht.

Transformationsregel 2 (nothing)



Lemma 4 *Es gilt die folgende Äquivalenz.*

```
nothing
```

$\Longleftrightarrow$

```
nothing
```

BEWEIS In diesem Fall offensichtlich.

3.4.4. Der Befehl halt

Esterelgrammatik

⟨*Halt*⟩ ::= 'halt'

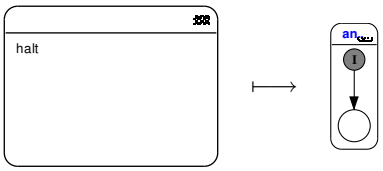
Beschreibung des Befehls

Der `halt`-Befehl pausiert für immer und terminiert niemals.

Beschreibung eines äquivalenten Makrozustandes

Eine unendliche Pause eines Makrozustandes wird durch einen direkten Übergang vom Initialen Zustand zu einem einfachen Zustand ohne ausgehende Transitionen erreicht.

Transformationsregel 3 (halt)



Lemma 5 *Es gilt die folgende Äquivalenz.*

$$\texttt{halt} \iff \begin{array}{l}\texttt{nothing;}\\ \texttt{halt}\end{array}$$

Beweis Folgt aus Lemma 1

3.4.5. Der Befehl pause

Esterelgrammatik

⟨*Pause*⟩ ::= 'pause'

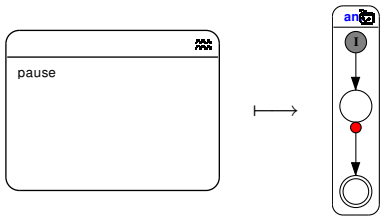
Beschreibung des Befehls

Der `pause`-Befehl unterbricht die Ausführung für eine Instanz, wobei laut *Berry* [5] `pause` äquivalent zu `await tick`.

Beschreibung eines äquivalenten Makrozustandes

Eine Unterbrechung der Ausführung wird dadurch erreicht, dass ein einfacher Zustand mit einer *strong abortion* die tick als *Trigger* besitzt, in einen einfachen, finalen Zustand überführt wird.

Transformationsregel 4 (pause)



Lemma 6 *Es gilt die folgende Äquivalenz.*

```
pause  ⟺  nothing;
          await case [tick] do
          nothing
          end await
```

Beweis Folgt aus Lemma 1, der Äquivalenz von `pause` zu `await tick` und der Äquivalenz von `await tick` zu `await case tick`.

3.4.6. Der Befehl abort

Esterelgrammatik

⟨*Abort*⟩ ::= <WeakAbort>
| 'abort' <Statement> 'when' <DelayExpression> ['do' <Statement>)] 'end' ['abort']
| 'abort' <Statement> 'when' <AbortCase> {<AbortCase>)} 'end' ['abort']

⟨*AbortCase*⟩ ::= 'case' <DelayExpression> ['do' <Statement>]

Beschreibung des Befehls

Der `abort`-Befehl bricht die Ausführung vom <Statement> ab, sobald eine <DelayExpression> erfüllt ist. Tritt keine <DelayExpression> auf und terminiert das <Statement>, so terminiert auch der `abort`-Befehl. Sollte zu einer ausgelösten <DelayExpression> ein 'do' <Statement> gehöhren, so wird dieser Teil in der Instanz des Abbruchs ausgeführt. Existieren verschiedene Abbruchbedinungen, so entspricht die textliche Reihenfolge der Priorität, wobei der textlich erste Fall, jener mit der höchsten Priorität ist.

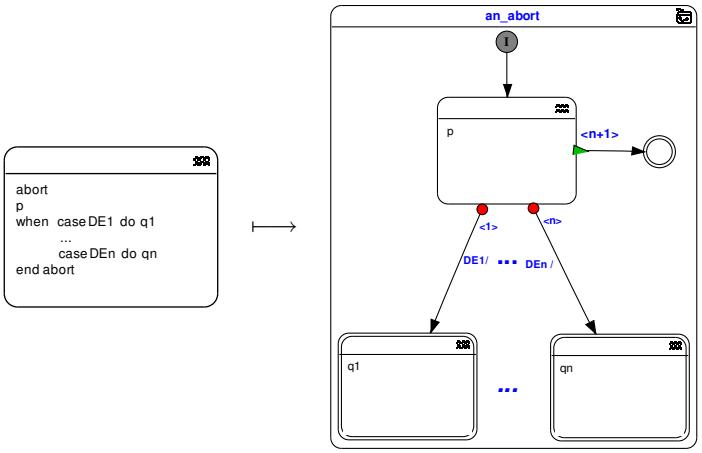
Beschreibung eines äquivalenten Makrozustandes

Um einen Makrozustand entsprechend der Semantik eines `abort`-Befehls abzubrechen, verwendet man *strong abortion*s mit den entsprechenden *Delayexpressions* als

Transitionsbeschriftungen mit der textlichen Reihenfolge als Priorität. Die *strong abortion*s führen zu finalen Makrozuständen, welche aus der Transformation der Statements enstehen.

Für den Fall einer normalen Terminierung des Befehls, welcher abgebrochen werden kann, führt eine *normal termination* zu einem einfachen, finalen Zustand.

Transformationsregel 5 (abort)



Transformationsregel 6 (abort in weak abort)

weak abort p when DE ⟼ *weak abort p when case DE do nothing*

Lemma 7 *Es gilt die folgende Äquivalenz.*

```
abort p
  case DE1 do run q1
   ...
  case DEn do run qn
end abort
```

⟺

```
nothing;
abort [p];
 nothing
when
case DE1 do
 nothing;
 [q1]
 ...
case DEn do
 nothing;
 [qn ]
end abort
```

Beweis Mit Hilfe des Lemma 1 ergibt sich die Äquivalenz.

3.4.7. Der Befehl assign

Esterelgrammatik

⟨*Assign*⟩ ::= <VariableIdentifier> ':=' <Expression>

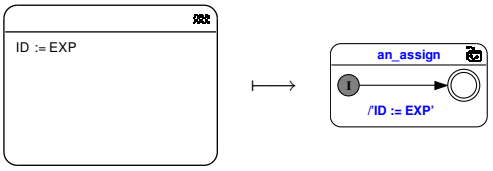
Beschreibung des Befehls

Dieser Befehl weist einer lokalen Variable eine <Expression> zu. Dazu müssen die Typen der <Expression> und der Variable dieselben sein.

Beschreibung eines äquivalenten Makrozustandes

Die Zuweisung wird als Aktion im Transitionsbeschriftung wie in Esterel angegeben.

Transformationsregel 7 (assign)



Lemma 8 *Es gilt die folgende Äquivalenz.*

```
ID := EXP   ⟺   nothing;
                ID := EXP
```

Beweis Hier folgt die Äquivalenz ebenfalls durch das Lemma 1.

3.4.8. Der Befehl await

Esterelgrammatik

⟨*Await*⟩ ::= 'await' <DelayExpression> ['do' <Statement> 'end' ['await']]
| 'await case' <DelayExpression> ['do' <Statement>]
{'case '<DelayExpression> ['do' <Statement>]}
'end' ['await']

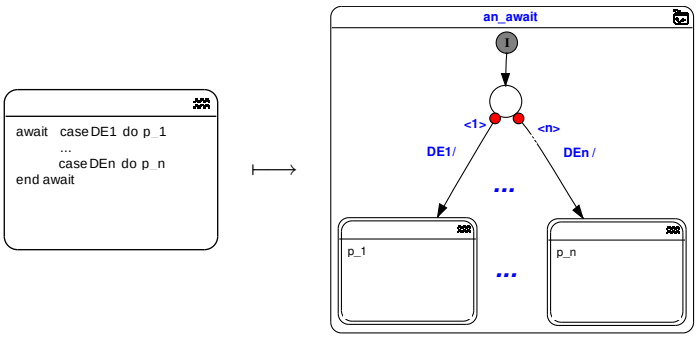
Beschreibung des Befehls

Der `await`-Befehl wartet auf das Eintreten einer <DelayExpression> und führt dann, wenn vorhanden, ein für diesen Fall vorgesehenes <Statement> aus. Stehen mehrere alternative *DelayExpressions* zur Auswahl, so wird zunächst die im Programmtext oberste überprüft und ggf. das <Statement> angewendet. Dies wird für alle *DelayExpressions* in der textlichen Reihenfolge wiederholt. Anschließend terminiert der `await`-Befehl. Die Schreibweise `await DE do p` ist nur eine Abkürzung für `await case DE do p` mit nur einem Fall.

Beschreibung eines äquivalenten Makrozustandes

Ein SyncChart verbleibt in einem Zustand, wenn dieser ein einfacher ist. Um diesen einfachen Zustand zu verlassen wird eine Transitionbenötigt. Hier fällt die Wahl auf eine *strong abortion*. Die als *Trigger* verwendete *Delayexpression* einer *strong abortion*, erzwingt eine Priorität der *strong abortion*, welche der Stelle *Delayexpression* in der textlichen Reihenfolge entspricht. Zudem ist das Ziel der *strong abortion* ein Makrozustand, welcher dem Verhalten vom <Statement> entspricht. Sollte in einem Fall der optionale Anweisungsteil nicht vorhanden sein, so wählt man als Ziel einen einfachen, finalen Zustand. Dieser entspricht der Umwandlung von `nothing`.

Transformationsregel 8 (await)



Transformationsregel 9 (await in await case)

await DE do p $\longmapsto$ *await case DE do p*

Lemma 9 *Es gilt die folgende Äquivalenz.*

```
await                          nothing;
 case DE1 do p_1               await
 ...                           case DE1 do
 case DEn do p_n                nothing;
end await                       [p_1 ]
                               case DEn do
                                nothing;
                                [p_n ]
                               end await
```

$\Longleftrightarrow$

BEWEIS Entfernt man die `nothing`-Befehle aufgrund des Lemma 1, so ergibt sich, dass auf beiden Seiten dieselben Programmstücke stehen. Diese müssen also auch semantisch äquivalent sein.

3.4.9. Der Befehl doupto

Esterelgrammatik

⟨*DoUpto*⟩ ::= 'do' <Statement> 'upto' <SignalExpression>

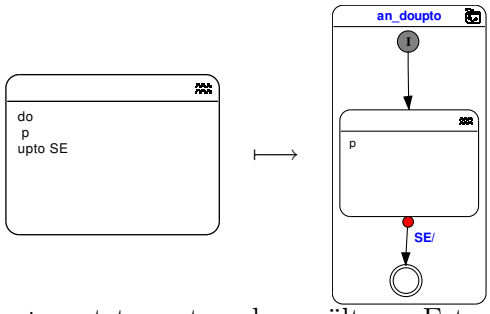
Beschreibung des Befehls

Es wird das <Statement> ausgeführt. Sollte dieses terminieren bevor die <SignalExpression> zutrifft, wird die weitere Ausführung angehalten und erst nach dem Auftreten der <SignalExpression> fortgesetzt. Tritt die <SignalExpression> während der Ausführung von <Statement> auf, so wird dieses sofort abgebrochen.

Beschreibung eines äquivalenten Makrozustandes

Das auszuführende <Statement> wird in einem Makrozustand dargestellt. Dieser Makrozustand kann nur über eine *strong abortion* mit der <SignalExpression> als Auslöser verlassen werden. Solange die <SignalExpression> nicht eintritt, kann der Makrozustand, ob terminiert oder aktiv, nicht verlassen werden.

Transformationsregel 10 (doupto)



Der Befehl `doupto` entstammt noch aus älteren Esterel Versionen und sollte nicht mehr verwendet werden. Im *Berry* [5] ist die folgende, äquivalente Ersetzung 3.1 angegeben.

```
do  p                ⟺    abort
upto SE                   p;halt
                          when SE
```

Abbildung 3.1.: Die Alternative zu `doupto`

Lemma 10 *Es gilt die folgende Äquivalenz.*

```
do p
upto S
```

$\Longleftrightarrow$

```
nothing;
abort
 [p];
 halt
when SE
do
 nothing
end abort
```

Beweis Die Behauptung folgt aus der Äquivalenz 3.1 und dem Lemma 1.

3.4.10. Der Befehl dowatching

Esterelgrammatik

⟨*DoWatching*⟩ ::= 'do' <Statement> 'watching'
<SignalExpression> ['timeout' <Statement> 'end timeout']

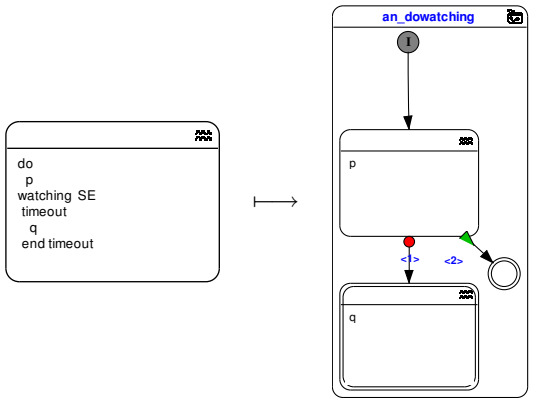
Beschreibung des Befehls

Beim Auftreten der <SignalExpression> wird das <Statement> sofort abgebrochen, ansonsten wird es normal ausgeführt. Terminiert das <Statement>, so terminiert auch der `dowatching` Befehl. Existiert ein timeout-Konstrukt, so wird dieses unmittelbar nach dem Auftreten der <SignalExpression> gestartet.

Beschreibung eines äquivalenten Makrozustandes

Das <Statement> wird durch einen Makrozustand dargestellt. Nach seiner Terminierung wird über eine *normal termination* zu einem einfachen, finalen Zustand übergegangen. Der Makrozustand wird durch eine *strong abortion* abgebrochen, sobald die <SignalExpression> eintritt. Die *strong abortion* endet an einem finalen Makrozustand, welches in einem timeout-Fall das entsprechende, auszuführende <Statement> oder ein `nothing` darstellt.

Transformationsregel 11 (dowatching)



Da der Befehl `dowatching` veraltet ist, sollte er entsprechend der Vorgaben aus *Berry* [5] ersetzt werden.

```
do p                    abort
watching SE              p
timeout          ⟺     when SE do
 q                       q
end timeout             end abort
```

Abbildung 3.2.: Die Alternative zu `dowatching`

Lemma 11 *Es gilt die folgende Äquivalenz.*

```
do  p                          nothing;
watching SE timeout            abort
  dotimeout                     [p];
end timeout                     nothing
                        ⟺     when [tick]
                               do
                                nothing;
                                [q]
                               end abort
                               }
```

Beweis Die Behauptung folgt aus der Äquivalenz 3.2 und dem Lemma 1.

3.4.11. Der Befehl emit

Esterelgrammatik

⟨*Emit*⟩ ::= 'emit' <SignalIdentifier> ['('Expression')']

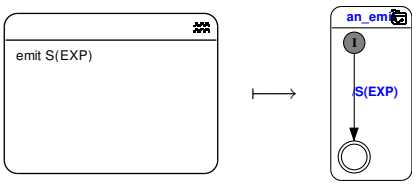
Beschreibung des Befehls

`emit` sendet ein Signal, wobei je nach Typ des Signals der Wert einer <Expression> beigefügt werden kann.

Beschreibung eines äquivalenten Makrozustandes

Um ein Signal zu senden, muss es in einer Transitionsbeschriftung als Aktion gesetzt werden. Da das Senden instantan ist, wird nur eine Transition benötigt. Diese braucht einen Start-und Endzustand den Initialen Zustand und einen einfachen, finalen Zustand.

Transformationsregel 12 (emit)



Lemma 12 *Es gilt die folgende Äquivalenz.*

emit S(EXP) ⟺ *emit S(EXP)*

Beweis Die linke und die rechte Seite sind offensichtlich äquivalent.

3.4.12. Der Befehl every

Esterelgrammatik

⟨*Every*⟩ ::= 'every' <DelayExpresion> 'do' <Statement> 'end' ['every']

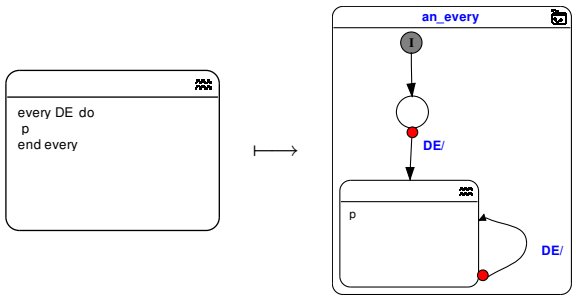
Beschreibung des Befehls

Das <Statement> wird erst ausgeführt, wenn die <DelayExpression> erfüllt ist. Anschließend wird das Statement bei jeder positiven Auswertung der <DelayExpression> neu aufgerufen, wobei der vorherige Aufruf zunächst abgebrochen wird. Sollte das <Statement> vor dem Auslösen der <DelayExpression> terminieren, so wird auf das Eintreten der <DelayExpression> gewartet. Das <Statement> darf nicht innerhalb einer Instanz terminieren, während die <DelayExpression> jederzeit erfüllt sein darf. Eine im *Berry* [5] angegebene Alternative ist in der Abbildung 3.3 dargestellt.

Beschreibung eines äquivalenten Makrozustandes

Da die Ausführung erst nach dem ersten Auftreten einer *Delayexpression* erfolgt, wird vom Startzustand direkt in einen einfachen Zustand übergegangen, welcher nur mittels einer *strong abortion* beim Zutreffen der *Delayexpression*, verlassen werden kann. Das Ziel der *strong abortion* ist ein Makrozustand, welcher als Start und Ziel einer *strong abortion*, wiederum mit der *Delayexpression* als Auslöser, dient. Dadurch entspricht das Verhalten dem von `every`. Der Makrozustand darf nicht in derselben Instanz gestartet werden und terminieren.

Transformationsregel 13 (every)




```
every DE do
   p
end every
```

$\Longleftrightarrow$

```
await DE ;
loop
  abort
   p;halt
 when DE
end loop
```

Abbildung 3.3.: Alternative zu `every`

Lemma 13 *Es gilt die folgende Äquivalenz.*

```
every DE do
 p
end every
```

$\Longleftrightarrow$

```
nothing;
await case [DE] do
nothing
end await ;
loop
 abort
  [p];
  halt
 when [DE]
 do
  nothing
 end abort
end loop
```

Beweis Die Behauptung folgt aus der Äquivalenz 3.3 und dem Lemma 1.

3.4.13. Der Befehl if

Esterelgrammatik

⟨*If*⟩ ::= 'if' <Expression> ['then' <Statment>] [<ElseIf> { <ElseIf> }] ['else' <Statement>] 'end' ['if']

⟨*ElseIf*⟩ ::= 'elsif' <Expression> ['then' <Statement>]

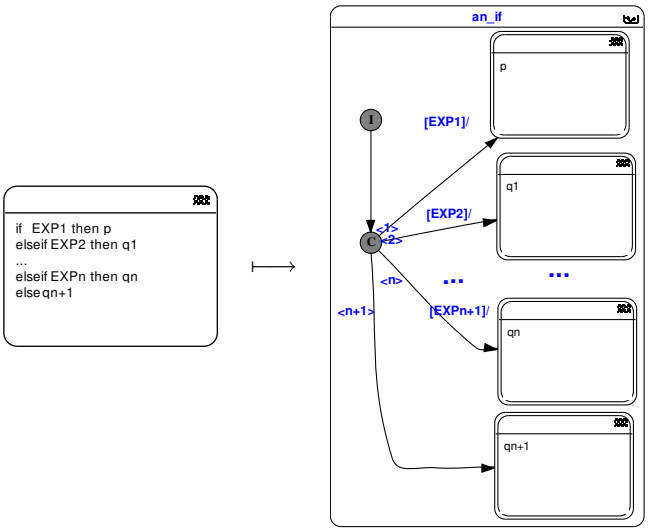
Beschreibung des Befehls

Der `if`-Befehl reagiert auf die Auswertung der <Expression>s. Sollte eine <Expression> zutreffen, so wird das zugehörige <Statement> sofort ausgeführt und keine weitere Auswertung der folgenden <Expression>s vorgenommen. Die Reihenfolge der Auswertung entpricht der textlichen Reihenfolge. Trifft keine `<Expression>` zu, so wird, falls angegeben, dass zu `else` gehörige `<Statement>` ausgeführt.

Beschreibung eines äquivalenten Makrozustandes

Eine äquivalente, grafische Darstellung kann man über den *Conditional*-Pseudozustand erreichen, indem man an die ausgehenden Transitionen die <Expression> als das Konditional angibt. Das Ziel der Transition ist ein Makrozustand, der das zugehörige <Statement> umsetzt. Da alle <Statement>s optional sind, verwendet man für nicht angegebene <Statement>s eine Umwandlung von `nothing`. Der `else`-Fall wird durch die erzwungende *default*-Transition dargestellt.

Transformationsregel 14 (if)



Lemma 14 *Es gilt die folgende Äquivalenz.*

```
if EXP1 then p
elsif EXP2 then q1
...
elsif EXPn then qn
else qn+1
end if
```

$\Longleftrightarrow$

```
nothing;
if EXP1 then
 nothing;
 [p]
elsif EXP2 then
 nothing;
 [q1]
 ...
elsif EXPn then
 nothing;
 [qn]
else
 nothing;
 [qn+1]
end if
```

Beweis Die Behauptung gilt aufgrund des Lemma 1.

3.4.14. Der Befehl localsignal

Esterelgrammatik

⟨*LocalSignal*⟩ ::= ‘signal’ <SignalDeclList> ‘in’ <Statement> [‘end signal’]

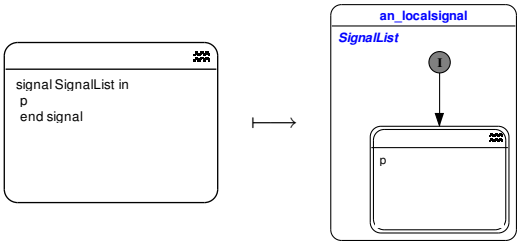
Beschreibung des Befehls

Der `localsignal` fügt die in der <SignalDeclList> stehenden Signale für das <Statement> hinzu. Bereits deklarierte Signale mit gleichen Namen werden verdeckt. Die lokalen Signale haben die gleichen Eigenschaften wie die Signale, die im Modulkopf deklariert worden sind, mit der Ausnahme, dass sie nur innerhalb des <Statement>s gültig sind.

Beschreibung eines äquivalenten Makrozustandes

Fügt man in einem Makrozustand die <SignalDeclList> entprechend ein, so kann man diese Signale im Makrozustand und allen, in der Hierariche folgenden Zuständen nutzen.

Transformationsregel 15 (localsignal)



Lemma 15 *Es gilt die folgende Äquivalenz.*

```
signal SignalList in
p
end signal
```
$\Longleftrightarrow$
```
signal SignalList
in
 nothing;
 [p]
end signal
```

Beweis Die Behauptung folgt aus dem Lemma 1.

3.4.15. Der Befehl localvariable

Esterelgrammatik

⟨*LocalVariable*⟩ ::= ‘var’ <VariableDeclarationList> ‘in’
<Statement>
‘end’ [‘var’]

⟨*VariableDeclarationList*⟩ ::= <Identifier> [‘:=’<Expression>] ‘:’ <TypeIdentifier>
{‘,’<Identifier> [‘:=’<Expression>] ‘:’ <TypeIdentifier>}

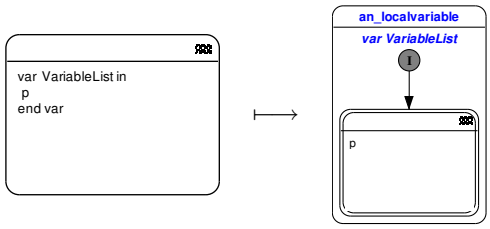
Beschreibung des Befehls

Der Befehl `var` fügt lokale Variablen für das <Statement> hinzu. Der Wert der lokalen Variablen kann, falls angegeben, mittels der <Expression> zugewiesen werden. Der Typ einer Variablen muss in jeden Fall angegeben werden. Es können beliebig viele Variablen deklariert werden.

Beschreibung eines äquivalenten Makrozustandes

Werden einem Makrozustand Variablen hinzugefügt, so können diese Variablen innerhalb des Makrozustandes und der in der Hierarchie untergeordneten Zustände verwendet werden.

Transformationsregel 16 (localvariable)



Lemma 16 *Es gilt die folgende Äquivalenz.*

```
var VariablenList in                    var VariableList
 p                                      in
end var                          ⟺      nothing;
                                        [p]
                                        end var
```

Beweis Die Behauptung folgt aus dem Lemma 1.

3.4.16. Der Befehl loop

Esterelgrammatik

⟨*Loop*⟩ ::= 'loop' <Statement> ['end loop']

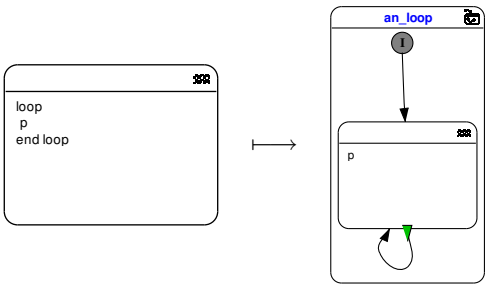
Beschreibung des Befehls

Eine Schleife darf nicht in der gleichen Instanz enden und eine Schleife terminiert nicht. Das Statement wird sofort nach der Terminierung wieder ausgeführt.

Beschreibung eines äquivalenten Makrozustandes

Diese Form einer Schleife ist durch eine *normal termination*darstellbar. Diese besitzt als Start und Ziel den Makrozustandes, welcher aus dem <Statement> abgeleitet wird. Dadurch wird der sofortige Wiedereintritt in den Makrozustand ermöglicht.

Transformationsregel 17 (loop)



Lemma 17 *Es gilt die folgende Äquivalenz.*

```
loop                          ⟺   nothing;
 p                                 loop
end loop                            [p
                                    ];
                                    nothing
                                  end loop
```

Beweis Die Behauptung gilt aufgrund von Lemma 1.

3.4.17. Der Befehl loopeach

Esterelgrammatik

⟨*LoopEach*⟩ ::= 'loop' <Statement> 'each' <DelayExpresion>

Beschreibung des Befehls

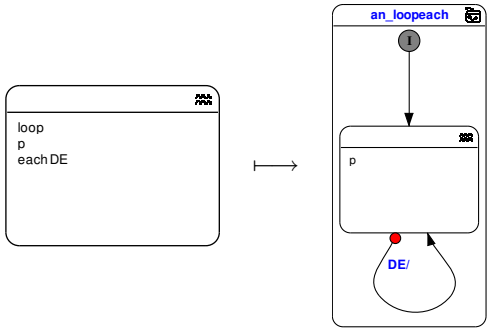
Die Schleife darf nicht in der gleichen Instanz enden und terminiert nicht. Das <Statement> wird am Anfang sofort gestartet, anschließend wird das <Statement> bei jeder positiven Auswertung der <DelayExpression> neu aufgerufen, wobei der vorherige Aufruf abgebrochen wird. Terminiert das <Statement> vor dem Auslösen der <DelayExpression>, so wird auf das Eintreten der <DelayExpression> gewartet. Eine altenative Schreibweise aus *Berry* [5] ist in der Abbildung 3.4 dargestellt.

Beschreibung eines äquivalenten Makrozustandes

Um das oben beschriebene Verhalten in einem SyncChart nachzubilden, wird eine *strong abortion* mit der <DelayExpression> als *Trigger* verwendet. Die *strong aborti-*

on beginnt und endet an dem Makrozustand, welcher das <Statement> darstellt. So wird der Makrozustand gegebenenfalls, abgebrochen und erneut ausgeführt, sobald die *DelayExpression* eintritt.

Transformationsregel 18 (loopeach)




```
loop                          loop
   p                            abort
each DE                          p;halt
                               when DE
                             end loop
```

Abbildung 3.4.: Alternative zu `loopeach`

Lemma 18 *Es gilt die folgende Äquivalenz.*

```
loop                          nothing;
 p                            loop
each DE                       abort
                               [p];
                               halt
                              when DE
                              do
                               nothing
                              end abort
                             end loop
```

Beweis Es gilt die folgende Äquivalenz.

3.4.18. Der Befehl parallel

Esterelgrammatik

⟨*Parallel*⟩ ::= (<NonParallel>
| <Parallel>) '||' <NonParallel>

⟨*NonParallel*⟩ ::= <AtomicStatement>
| <Sequence>

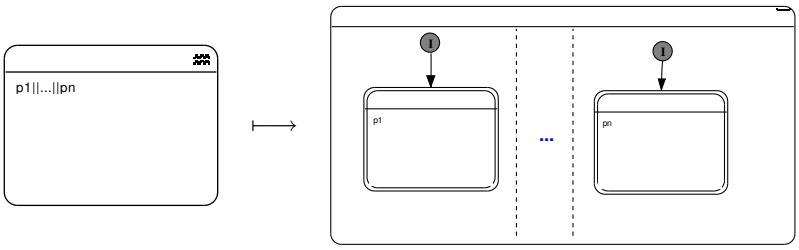
Beschreibung des Befehls

Durch das || Kommando können beliebig viele nicht-parallele Befehle p_i $i \in 0, \ldots, n$ unmittelbar parallel ausgeführt werden. Das || Komando terminiert erst, wenn alle p_i terminiert sind.

Beschreibung eines äquivalenten Makrozustandes

Die einzelnen <NonParallel>-Befehle werden jeweils in einen Makrozustand umgewandelt. Diese Makrozustände werden in einen parallelen Makrozustand eingefügt.

Transformationsregel 19 (parallel)



Lemma 19 *Es gilt die folgende Äquivalenz.*

```
p1 || .. || pn   ⟺   [ nothing;
                       [p1]
                     ||
                     ...
                     ||
                      nothing;
                       [pn ]
                     ]
```

Beweis Die Behauptung folgt unter Einbeziehung von Lemma 1.

3.4.19. Der Befehl present

Esterelgrammatik

⟨*Present*⟩ ::= 'present'
'end' ['present']

⟨*Case*⟩ ::= 'case' <PresentEvent> ['do' <Statement>]

⟨*PresentEvent*⟩ ::= '[' <SignalExpression> ']'
| <SignalExpression>

Beschreibung des Befehls

Der `present`-Befehl testet, ob eine `<SignalExpression>` erfüllt ist oder nicht. Dabei stehen dabei zwei Varianten zur Verfügung.

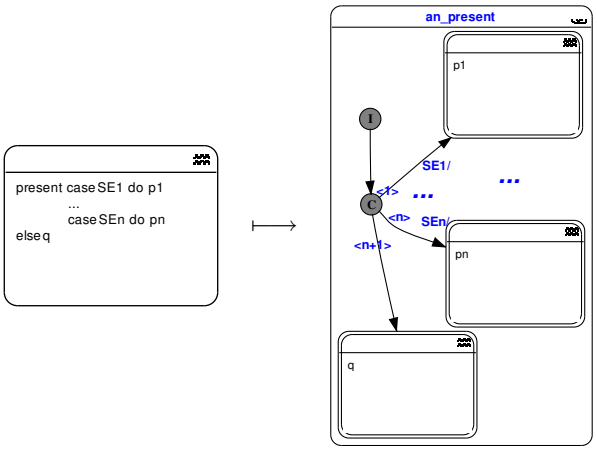
- `present SE then p else q end present`
- `present case SE1 do p1 ... case SEn do pn else q end present`

Die erste Variante ist nur eine abkürzende Schreibweise der zweiten für den Fall, dass nur eine Bedinung zu überprüfen ist. Ist ein Signalausdruck wahr, so wird das entsprechende <Statement> sofort ausgeführt und nach dessen Terminierung der `present`- Befehl ebenfalls terminiert. Die Reihenfolge der Überprüfung enspricht der textlichen Reihenfolge. Sollte keine der Überprüfungen ein positives Ergebnis liefern, wird ein möglicher `else`-Zweig ausgeführt.

Beschreibung eines äquivalenten Makrozustandes

Für die Darstellung im SyncChart sind die gleichen Überlegungen anzustellen, wie im Falle von *if* 3.4.13, nur werden die `<SignalExpression>`s als *Trigger* nicht als Kondition verwendet.

Transformationsregel 20 (present)



Transformationsregel 21 (then in case)

then $\mapsto$ *case*

Lemma 20 *Es gilt die folgende Äquivalenz.*

```
present case SE1 do p1                    nothing;
        ...                               present
        case SEn do pn                    case SE1 do
        else q                             nothing;
end present                                [p1]
                                          ...
                                  ⟺      case SEn do
                                           nothing;
                                           [pn]
                                          else
                                           nothing;
                                           [q]
                                          end present
```

Beweis Die Behauptung folgt aufgrund von Lemma 1 und Lemma 2.

3.4.20. Der Befehl call

Esterelgrammatik

⟨*Call*⟩ ::= ‘call’ <ProcedureIdentifier> ‘(’ { <VariableIdentifier> } >‘)’‘(’{ <Expression> }‘)’

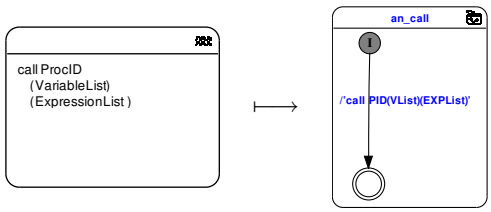
Beschreibung des Befehls

Der `call`-Befehl ruft eine Prozedur auf, welche extern definiert wird. Dieser Aufruf muss in derselben Instanz enden, in welcher er gestartet wurde. Dabei können mehrere Variablen übergeben werden. Die Werte dieser Variablen können durch die Prozedur verändert werden. Zusätzlich können Ausdrücke übergeben werden. Die Reihenfolge der Typen der Parameter muss mit der in der Deklaration zu Beginn des Moduls identisch sein.

Beschreibung eines äquivalenten Makrozustandes

Der Aufruf wird als Aktion einer Transitionsbeschriftung hinzugefügt.

Transformationsregel 22 (call)



Lemma 21 *Es gilt die folgende Äquivalenz.*

$$\textit{call PID(VList)(EXPList)} \quad \Longleftrightarrow \quad \begin{array}{l}\textit{nothing;}\\ \textit{call PID(VList)(EXPList)}\end{array}$$

Beweis Die Behauptung folgt aus dem Lemma 1.

3.4.21. Der Befehl sequence

Esterelgrammatik

⟨*Sequence*⟩ ::= <SequenceWithoutTerminator> [';']

⟨*SequenceWithoutTerminator*⟩ ::= <AtomicStatement>';'<AtomicStatement>
| <SequenceWithoutTerminator>';'<AtomicStatement>

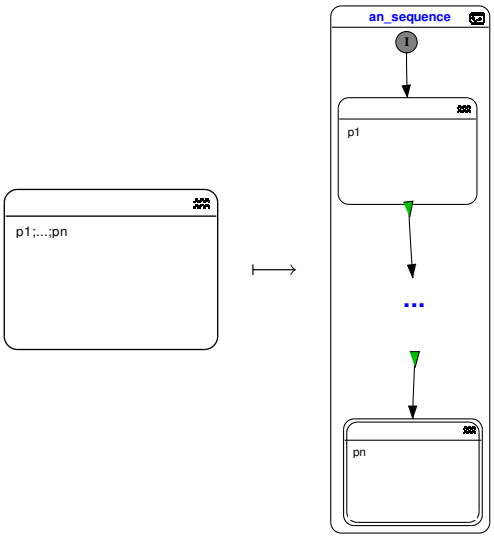
Beschreibung des Befehls

Eine Sequenz von <AtomicStatement>;...;<AtomicStatement> wird sofort ausgeführt. Terminiert ein <AtomicStatement>, so wird sofort das Folgende ausgeführt. Terminiert das letzte <AtomicStatement>, so terminiert die Sequenz ebenfalls. Wird innerhalb der Sequenz eine, der Sequenz übergeordnete, *Trap* ausgelöst, so werden nachfolgende <AtomicStatement>s nicht mehr ausgeführt.

Beschreibung eines äquivalenten Makrozustandes

Die einzelnen Makrozustände, welche ein <AtomicStatement> darstellen, werden mit Hilfe von *normal terminations* in der textlichen Reihenfolge der <AtomicStatement>s miteinander verbunden. Die Ausführung im Falle einer ausgelösten *Trap* wird in dem entsprechenden Makrozustand unterbrochen, da dieser in dem Fall nicht terminiert.

Transformationsregel 23 (sequence)



Lemma 22 *Es gilt die folgende Äquivalenz.*

$$p1;\ldots;pn \iff \begin{array}{l} nothing; \\ [p1]; \\ \ldots \\ nothing; \\ [pn]; \end{array}$$

Beweis Aufgrund von Lemma 1 ist die Behauptung wahr.

3.4.22. Der Befehl suspend

Esterelgrammatik

⟨*Suspend*⟩ ::= 'suspend' <Statement> 'when' ['immediate'] <SignalExpression>

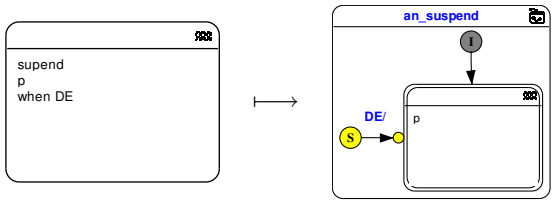
Beschreibung des Befehls

Der `suspend`-Befehl unterbricht die Ausführung des <Statement>s, solange die <SignalExpression> gültig ist.

Beschreibung eines äquivalenten Makrozustandes

Nur für diesen Befehl wird der *Suspend*-Pseudozustand mit einer *Suspension* zum Makrozustand des umgewandelten <Statement>s benutzt. Dabei wird die <Signal-Expression> inklusive des eventuellen `immediate` als *Trigger* verwendet.

Transformationsregel 24 (suspend)



Lemma 23 *Es gilt die folgende Äquivalenz.*

```
suspend
 p
when DE
```
$\Longleftrightarrow$
```
nothing;
suspend
 [p]
when DE
```

Beweis Die Behauptung trifft aufgrund des Lemma 1 zu.

3.4.23. Der Befehl sustain

Esterelgrammatik

⟨*Sustain*⟩ ::= 'sustain' <SignalIdentifier> ['(' <Expression>')']

Beschreibung des Befehls

Der `sustain`-Befehl sendet in jeder Instanz das Signal <SignalIdentifier>. Kann das Signal einen Wert beinhalten und ist es in Form der <Expression> angegeben, so wird das Signal entsprechend gesetzt. Eine alternative Schreibweise aus *Berry* [5] ist in der Abbildung 3.5 angegeben.

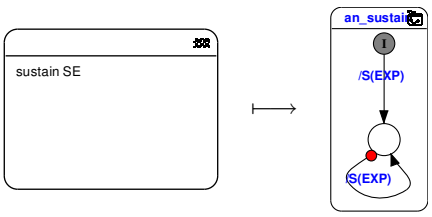
Beschreibung eines äquivalenten Makrozustandes

Ein einfacher Zustand wird durch eine *strong abortion* in jeder Instanz abgebrochen, wobei <SignalIdentifier> ['(' <Expression>')' gesendet wird. Das Ziel der *strong abortion* ist wieder der einfache Zustand. Zusätzlich muß beim Übergang vom Initialzustand zum einfachen Zustand das Signal <SignalIdentifier> ['(' <Expression>')' gesendet werden, da es sonst in der ersten Instanz nicht gesendet werden würde.

```
sustain S(EXP)          ⟺   loop
                             emit S(EXP)
                             each tick
```

Abbildung 3.5.: Alternative zu `sustain`

Transformationsregel 25 (sustain)



Lemma 24 *Es gilt die folgende Äquivalenz.*

```
sustain S(EXP)
```
$\Longleftrightarrow$
```
emit S(EXP);
loop
 await case [tick] do
 emit S(EXP)
 end await
end loop
```

Beweis Um den Beweis übersichtlich zu halten, wird an dieser Stelle mit einer Ableitung begonnen, welche im Verlauf des Beweises mehrfach benötigt wird.

Lemma 25
Es gilt die Abbleitung, die ich mit Hilf1 *abbkürzen wird:*

$$((\{tick \Rightarrow !S_1;2\};\ S_1?S(EXP),0)\ )\backslash\ S_1)^* \xrightarrow[E]{\emptyset,1} (\{0;tick\cdot \Rightarrow (!S_1;2)\};\ S_1?S(EXP),0)\backslash\ S_1;\ ((\{tick \Rightarrow !S_1;2\};\ S_1?S(EXP),0)\ )\backslash\ S_1)^*$$

Beweis
Es gilt:

(unit delay) 1	$\xrightarrow[E]{\emptyset,1}$	0
(sequence1)1;tick$\cdot \Rightarrow$(!S_1;2)	$\xrightarrow[E]{\emptyset,1}$	0;tick$\cdot \Rightarrow$(!S_1;2)
tick$\Rightarrow$(!S_1;2)	$\xrightarrow[E]{\emptyset,1}$	0;tick$\cdot \Rightarrow$(!S_1;2)
(sequence1)1;tick$\cdot \Rightarrow$(!S_1;2)	$\xrightarrow[E]{\emptyset,1}$	0;tick$\cdot \Rightarrow$(!S_1;2)
(trap2) {tick$\Rightarrow$(!S_1;2)}	$\xrightarrow[E]{\emptyset,1}$	{0;tick$\cdot \Rightarrow$(!S_1;2)}
(seq1) {tick$\Rightarrow$!(S$_1$;2) }; S_1?S(EXP),0	$\xrightarrow[E]{\emptyset,1}$	{0;tick$\cdot \Rightarrow$(!S_1;2)}; S_1?S(EXP),0
(sig-) ({tick$\Rightarrow$!S_1;2 }; S_1?S(EXP),0))\ S_1	$\xrightarrow[E]{\emptyset,1}$	({0;tick$\cdot \Rightarrow$(!S_1;2)}; S_1?S(EXP),0)\ S_1
(loop) ((({tick$\Rightarrow$!S_1;2 }; S_1?S(EXP),0))\ S_1)*	$\xrightarrow[E]{\emptyset,1}$	({0;tick$\cdot \Rightarrow$(!S_1;2)}; S_1?S(EXP),0)\ S_1; ((({tick$\Rightarrow$!S_1;2 }; S_1?S(EXP),0))\ S_1)*

Es ist nun zu zeigen, dass das Signal S(EXP) in jeder Instanz gesendet wird, um im Verhalten `sustain S(EXP)` zu gleichen. Für die erste Instanz gilt:

(emit) !S(EXP) $\xrightarrow[E]{\text{S(EXP)},0}$ 0 (Lemma 25) *Hilf1*

(seq2)) !S(EXP); (tick$\not\Rightarrow$!S(EXP))* $\xrightarrow[E]{S(EXP),1}$ ({0;tick$\cdot \Rightarrow$(!S_1;2)}; S_1?S(EXP),0)\ S_1; ((({tick$\Rightarrow$!S_1;2 }; S_1?S(EXP),0))\ S_1)*

Das Signal S(EXP) wird emitiert. Somit ist zu zeigen, dass S(EXP) auch in den folgenden Inztanzen vorliegt. Für die zweite Instanz folgt dann:

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
(\mathrm{emit})!\mathsf{S}_1 \xrightarrow[E]{S_1,0} 0 \qquad (\mathrm{exit})\ 2 \xrightarrow[E]{\emptyset,2} 0
}{
(\mathrm{seq2})!\mathsf{S}_1;2 \xrightarrow[E]{S_1,2} 0
}
}{
(\mathrm{shift})\uparrow(!\mathsf{S}_1;2) \xrightarrow[E]{S_1,3} \uparrow(0)
}
}{
(\mathrm{present}+)(\mathrm{tick})?(\uparrow(!\mathsf{S}_1;2),1) \xrightarrow[E]{S_1,3} \uparrow(0)
}
}{
(\mathrm{loop})((\mathrm{tick})?(\uparrow(!\mathsf{S}_1;2),1))^* \xrightarrow[E]{S_1,3} \uparrow (0);(\uparrow(!\mathsf{S}_1;2),1))^*
}
}{
(\mathrm{trap2})\{((\mathrm{tick})?(\uparrow(!\mathsf{S}_1;2),1))^*\ \} \xrightarrow[E]{S_1,2} \{\uparrow(0);(\uparrow(!\mathsf{S}_1;2),1))^*\}
}
}{
(\mathrm{trap})(\mathrm{nothing}))\ \{0;\mathsf{tick}\cdot \Rightarrow(!\mathsf{S}_1;2)\} \xrightarrow[E]{S_1,0} 0
}
\qquad
\dfrac{
(\mathrm{emit})!\mathsf{S(EXP)} \xrightarrow[E]{S(EXP),0} 0
}{
(\mathrm{present}+)\mathsf{S}_1?\mathsf{S(EXP)},0) \xrightarrow[E]{S(EXP),0} 0
}
}{
(\mathrm{seq2})\ \{0;\mathsf{tick}\cdot \Rightarrow(!\mathsf{S}_1;2)\}\ ;\ \mathsf{S}_1?\mathsf{S(EXP)},0 \xrightarrow[E]{S_1 * S(EXP),0} 0
}
}{
(\mathrm{sig}+)\ (\{0;\mathsf{tick}\cdot \Rightarrow(!\mathsf{S}_1;2)\};\ \mathsf{S}_1?\mathsf{S(EXP)},0)\backslash\ \mathsf{S}_1 \xrightarrow[E]{S(EXP),0} (0)\backslash\ \mathsf{S}_1
}
\quad (\text{Lemma 25})\ \textit{Hilf1}
}{
\begin{array}{c}(\mathrm{seq2})\ (\{0;\mathsf{tick}\cdot \Rightarrow(!\mathsf{S}_1;2)\};\ \mathsf{S}_1?\mathsf{S(EXP)},0)\backslash\ \mathsf{S}_1;\\ (((\{\mathsf{tick}\Rightarrow!\mathsf{S}_1;2\ \};\ \mathsf{S}_1?\mathsf{S(EXP)},0)\ )\backslash\ \mathsf{S}_1)^*\end{array}
\xrightarrow[E]{S(EXP),1}
\begin{array}{c}(\{0;\mathsf{tick}\cdot \Rightarrow(!\mathsf{S}_1;2)\};\ \mathsf{S}_1?\mathsf{S(EXP)},0)\backslash\\ (((\{\mathsf{tick}\Rightarrow!\mathsf{S}_1;2\ \};\ \mathsf{S}_1?\mathsf{S(EXP)},0)\ )\backslash\ \mathsf{S}_1)^*\end{array}
}
$$

Da nach der zweiten Instanz die linke und die rechte Seite übereinstimmen, wird sich dieses Verhalten immer wiederholen, also wird insbesondere das Signal (S(EXP)) in jeder Instanz emitiert. Somit ist die Behauptung wahr.

3.4.24. Der Befehl trap

Esterelgrammatik

⟨*Trap*⟩ ::= 'trap' <ExceptionDeclaration> {<ExceptionDeclaration>} 'in'
<Statement> [<ExceptionHandler> {<ExceptionHandler>}] 'end'
['trap']

⟨*ExceptionDeclaration*⟩ ::= <ExceptionIdentifier> [[':='<Expressiom>]
':' <ChannelType>]

⟨*ExceptionHandler*⟩ ::= 'handle' <ExceptionEvent> 'do' <Statement>

⟨*ExceptionEvent*⟩ ::= <ExceptionIdentifier>
| '('<ExceptionEvent>')'
| 'not'<ExceptionEvent>
| <ExceptionEvent> ('and' | 'or') <ExceptionEvent>

Beschreibung des Befehls

Ein `trap`-Befehl bricht die Ausführung des <Statement>s ab, sobald ein <ExceptionEvent> auftritt. Tritt keines auf, wird das <Statement> normal durchlaufen. Optional kann man für verschiedene Ausnahmefälle bestimmte Programmstücke ausführen. Treffen mehrere Ausnahmefälle zu, so werden die entsprechenden Programmstücke parallel durchlaufen.

Beschreibung eines äquivalenten Makrozustandes

Es muss zunächst ein lokales Signal als Ersatz für die in SyncChart nicht vorhandenen Ausnahmesignale, für jede <ExceptionDeclaration> sowie ein neues Signal zum Anhalten der Ausführung der `trap` eingeführt werden. Dieses Signal wird im Folgenden mit *traphalt* bezeichnen. Die Initiale Transition geht dann zum Makrozustand des <Statement>s über. Von diesem Makrozustand führt eine *normal termination* in einen finalen, einfachen Zustand, falls keine Ausnahme eintritt, eine *weak abortion* in einen einfachen Zustand mit dem Signal traphalt als *Trigger* und eine *weak abortion* mit dem *Trigger* der Oder-Verknüpften <ExceptionIdentifier>, welcher zusätzlich auch `immediate` ist, in einen finalen, parallelen Makrozustand. In diesem Makrozustand werden parallel die einzelnen <ExceptionEvents> auf ihr Vorhandensein überprüft und gegebenenfalls das zugehörige <Statement> ausgeführt. Sollte kein <ExceptionHandler> deklariert worden sein, so wird der finale parallele Makrozustand durch einen einfachen, finalen Zustand ersetzt.

Wie beschrieben, wird ein `trap` umgewandelt, indem ein schwacher Abbruch des eingeschlossenen Befehls ausgelöst wird. Dadurch ergeben sich die folgenden Problematiken:

- Priorität der übergeordneten *Traps*,

- Priorität gegenüber eines normalen schwachen Abbruchs,
- Anhalten von Sequenzen,

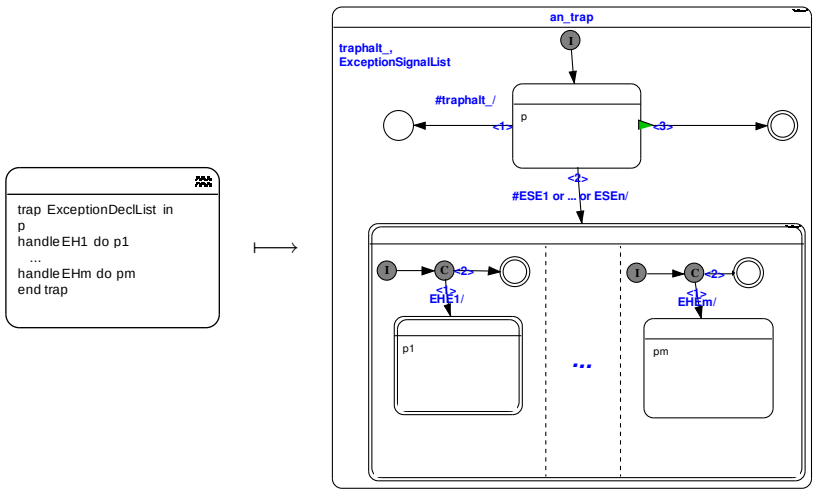
Der erste Punkt wird mit Hilfe des traphalt-Signals gelöst, indem dieses Signal beim Auslösen eines Ausnahmesignals einer priorisierten `trap` durch `exit` 3.4.25 zusätzlich gesendet wird. Daher werden neben dem Ausnahmesignal alle traphalt-Signale der Umwandlung eines `exit`-Befehls gesendet, die in der Hierarchie zwischen der ausgelösten *Trap* und dem *Exit* stehen.

Bricht die Umwandlung einer `trap` aufgrund des entsprechenden traphalt die Ausführung ihres zugehörigen Befehls ab, so wird in einen einfachen Zustand übergegangen, der keine ausgehenden Transitionen besitzt. Die Ausführung wird also angehalten.

Der zweite Punkt kann ähnlich wie der erste gelöst werden, dieses genauer im Abschnitt 3.4.26 beschrieben.

Die Problemstellung des letzten Punktes wird dadurch gelöst, dass zunächst die ersten beiden Punkte erfüllt werden, also Komponenten, die den Programmfluß verändern können, angehalten werden und dadurch, dass die Umwandlung eines `exit`-Befehls nicht terminiert, Befehle die dem `exit` folgen, nicht mehr ausgeführt werden.

Transformationsregel 26 (trap)



Zunächst werden Abkürzungen und Hilfslemmata, die für diesen Beweis gelten, definieren.

traphalt	th
ESE_1 or ... or ESE_n	ESE
sc_weak1	w1
sc_go_state_20	g20
sc_go_state_23	g23
$\star$	beliebig

Lemma 26 (H1)
Seien S,HS *Signale und* $r := (\mathsf{S}?(\uparrow(!\mathsf{HS};2);2),1)^*$. *Dann gilt*

(1) $\{r\} \xrightarrow[E]{\{HS\},2} \{\uparrow 0;2;r\}$ *für* $S \in E$

(2) $\{r\} \xrightarrow[E]{\emptyset,0} \{r\}$ *für* $S \notin E$

Beweis Es gilt:

(1) $S \in E$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(\text{emit})!\mathsf{HS} \xrightarrow[E]{\{HS\},0} 0 \qquad (\text{exit})2 \xrightarrow[E]{\emptyset,2} 0}{(\text{seq2})!\mathsf{HS};2 \xrightarrow[E]{\{HS\},2} 0}}{(\text{shift})\uparrow(!\mathsf{HS};2) \xrightarrow[E]{\{HS\},3} \uparrow 0}}{(\text{seq1})\uparrow(!\mathsf{HS};2);2 \xrightarrow[E]{\{HS\},3} \uparrow 0;2}}{(\text{present+})\mathsf{S}?(\uparrow(!\mathsf{HS};2);2),1 \xrightarrow[E]{\{HS\},3} \uparrow 0;2}}{(\text{loop})(\mathsf{S}?(\uparrow(!\mathsf{HS};2);2),1)^* \xrightarrow[E]{\{HS\},3} \uparrow 0;2;r}}{(\text{trap2})\ \{(\mathsf{S}?(\uparrow(!\mathsf{HS};2);2),1)^*\} \xrightarrow[E]{\{HS\},2} \{\uparrow 0;2;r\}}$$

(2) $S \notin E$

$$\dfrac{\dfrac{\dfrac{(\text{unit delay})1 \xrightarrow[E]{\emptyset,0} 0}{(\text{present-})\mathsf{S}?(\uparrow(!\mathsf{HS};2);2),1 \xrightarrow[E]{\emptyset,0} 0}}{(\text{loop})(\mathsf{S}?(\uparrow(!\mathsf{HS};2);2),1)^* \xrightarrow[E]{\emptyset,0} 0;r}}{(\text{trap})\ \{(\mathsf{S}?(\uparrow(!\mathsf{HS};2);2),1)^*\} \xrightarrow[E]{\emptyset,0} \{0;r\}}$$

Mit Lemma 1 folgt die Behauptung.

Lemma 27 (H2) *Seien* S_1,S_2,HS_1,HS_2 *Signale,* $r_1 := (\mathsf{S}_1?(\uparrow(!\mathsf{HS}_1;2);2),1)^*$ *und* $r_2 := (\mathsf{S}_2?(\uparrow(!\mathsf{HS}_2;2);2),1)^*$. *Dann gilt:*

(1) $\mathtt{p}$ *terminiert nicht, also* $k_{r_3} = 1$ *und* S_1,$\mathsf{S}_2 \notin E_p$. *Dann gilt*
$\{\uparrow\mathtt{p};2 \mid \{r_1\} \mid \{r_2\}\} \xrightarrow[E]{E_p,1} \{\uparrow\mathtt{p'};2 \mid \{r_1\} \mid \{r_2\}\}$

(2) $\mathtt{p}$ *terminiert,* S_1,$\mathsf{S}_2 \in E_p$. *Dann gilt* $\{\uparrow\mathtt{p};2 \mid \{r_1\} \mid \{r_2\}\} \xrightarrow[E]{E_p,0} 0$

(3) S_1,$\mathsf{S}_2 \in E_p$, *dann gilt* $\{\uparrow\mathtt{p};2 \mid \{r_1\} \mid \{r_2\}\} \xrightarrow[E]{E_p*\{HS_1\}*\{HS_2\},0} 0$

(4) $\mathsf{S}_1 \in E_p$ *und* $\mathsf{S}_2 \notin E_p$*, dann gilt* $\{\uparrow \mathtt{p};2 \mid \{r_1\} \mid \{r_2\}\} \xrightarrow[E]{E_p * \{HS_1\},0} 0$

(5) $\mathsf{S}_1 \notin E_p$ *und* $\mathsf{S}_2 \in E_p$*, dann gilt* $\{\uparrow \mathtt{p};2 \mid \{r_1\} \mid \{r_2\}\} \xrightarrow[E]{E_p * \{HS_2\},0} 0$

BEWEIS Da `p` aus dem SyncChart folgt, enthält `p` kein `exit` für einen priorisierten `trap`-Befehl. Somit folgt für $\mathtt{p} \xrightarrow[E]{E_p,k_p} \mathtt{p'}$, dass $k_p < 2$ ist. Aus diesem Grund entspricht der *Completion-Code* von $\uparrow \mathtt{p}$ k_p. Terminiert `p`, so folgt für $r_3 := \uparrow \mathtt{p};2$ $r_3 \xrightarrow[E]{E_p,k_{r_3}} 0$ mit $k_{r_3} = 2$.

Es gilt:

(1) `p` terminiert nicht, also $k_{r_3} = 1$ und $\mathsf{S}_1, \mathsf{S}_2 \notin E_p$.

Es gilt:

$$\frac{\dfrac{\uparrow \mathtt{p};2 \xrightarrow[E]{E_p,1} \uparrow \mathtt{p'};2 \qquad (\mathrm{H1})\{r_1\} \xrightarrow[E]{\emptyset,0} \{r_1\} \qquad (\mathrm{H1})\{r_2\} \xrightarrow[E]{\emptyset,0} \{r_2\}}{(\mathrm{parallel}) \uparrow \mathtt{p};2 \mid \{r_1\} \mid \{r_2\} \xrightarrow[E]{E_p,1} \uparrow \mathtt{p'};2 \mid \{r_1\} \mid \{r_2\}}}{(\mathrm{trap2})\ \{\uparrow \mathtt{p};2 \mid \{r_1\} \mid \{r_2\}\} \xrightarrow[E]{E_p,1} \{\uparrow \mathtt{p'};2 \mid \{r_1\} \mid \{r_2\}\}}$$

(2) `p` terminiert, $\mathsf{S}_1, \mathsf{S}_2 \in E_p$.

Dann gilt:

$$\frac{\dfrac{\uparrow \mathtt{p};2 \xrightarrow[E]{E_p,2} 0 \qquad (\mathrm{H1})\{r_1\} \xrightarrow[E]{\emptyset,0} \{r_1\} \qquad (\mathrm{H1})\{r_2\} \xrightarrow[E]{\emptyset,0} \{r_2\}}{(\mathrm{parallel}) \uparrow \mathtt{p};2 \mid \{r_1\} \mid \{r_2\} \xrightarrow[E]{E_p,2} 0 \mid \{r_1\} \mid \{r_2\}}}{(\mathrm{trap1})\ \{\uparrow \mathtt{p};2 \mid \{r_1\} \mid \{r_2\}\} \xrightarrow[E]{E_p,0} 0}$$

(3) $\mathsf{S}_1, \mathsf{S}_2 \in E_p$.

Dann gilt:

$$\frac{\dfrac{\uparrow \mathtt{p};2 \xrightarrow[E]{E_p,k_{r_3}<3} \star \quad (\mathrm{H1})\{r_1\} \xrightarrow[E]{\{HS_1\},2} \{\uparrow 0;2;r_1\} \quad (\mathrm{H1})\{r_2\} \xrightarrow[E]{\{HS_2\},2} \{\uparrow 0;2;r_2\}}{(\mathrm{parallel}) \uparrow \mathtt{p};2 \mid \{r_1\} \mid \{r_2\} \xrightarrow[E]{E_p * \{HS_1\} * \{HS_2\},2} \star \mid \{\uparrow 0;2;r_1\} \mid \{\uparrow 0;2;r_2\}}}{(\mathrm{trap1})\ \{\uparrow \mathtt{p};2 \mid \{r_1\} \mid \{r_2\}\} \xrightarrow[E]{E_p * \{HS_1\} * \{HS_2\},0} 0}$$

(4) $\mathsf{S}_1 \in E_p$ und $\mathsf{S}_2 \notin E_p$.

Dann gilt:

$$\frac{\dfrac{\uparrow \mathtt{p};2 \xrightarrow[E]{E_p,k_{r_3}<3} \star \qquad (\mathrm{H1})\{r_1\} \xrightarrow[E]{\{HS_1\},2} \{\uparrow 0;2;r_1\} \qquad (\mathrm{H1})\{r_2\} \xrightarrow[E]{\emptyset,0} \{r_2\}}{(\mathrm{parallel}) \uparrow \mathtt{p};2 \mid \{r_1\} \mid \{r_2\} \xrightarrow[E]{E_p * \{HS_1\},2} \star \mid \{\uparrow 0;2;r_1\} \mid \{r_2\}}}{(\mathrm{trap1})\ \{\uparrow \mathtt{p};2 \mid \{r_1\} \mid \{r_2\}\} \xrightarrow[E]{E_p * \{HS_1\},0} 0}$$

(5) $\mathsf{S}_1 \notin E_p$ und $\mathsf{S}_2 \in E_p$.

Dann gilt

$$\frac{\uparrow\mathtt{p};2 \xrightarrow[E]{E_p, k_{r_3}<3} \star \qquad (\mathrm{H1})\{r_1\} \xrightarrow[E]{\emptyset,0} \{r_1\} \qquad (\mathrm{H1})\{r_2\} \xrightarrow[E]{\{HS_2\},2} \{\uparrow 0;2;r_2\}}{(\mathrm{parallel})\uparrow\mathtt{p};2 \mid \{r_1\} \mid \{r_2\} \xrightarrow[E]{E_p * \{HS_2\},2} \star \mid \{r_1\} \mid \{\uparrow 0;2;r_2\}}$$

$$\frac{(\mathrm{parallel})\uparrow\mathtt{p};2 \mid \{r_1\} \mid \{r_2\} \xrightarrow[E]{E_p * \{HS_2\},2} \star \mid \{r_1\} \mid \{\uparrow 0;2;r_2\}}{(\mathrm{trap1})\ \{\uparrow\mathtt{p};2 \mid \{r_1\} \mid \{r_2\}\} \xrightarrow[E]{E_p * \{HS_2\},0} 0}$$

Also gelten die Behauptungen.

Lemma 28 (H3) *Seien S_1,S_2, HS_1,HS_2,HS_3 Signale*

(1) Ist $S_1 \in E$ so gilt:
S_1,S_2*?(!*HS_2*;!*HS_1*),(!*HS_2*;!* HS_3*)* ,0 $\xrightarrow[E]{\{HS_2,HS_1\},0}$ 0.

(2) Ist $S_2 \in E$ $S_1 \notin E$
Es gilt:
S_1,S_2*?(!*HS_2*;!*HS_1*),(!*HS_2*;!* HS_3*)* ,0 $\xrightarrow[E]{\{HS_2,HS_3\},0}$ 0

(3) Ist S_1, $S_2 \notin E$ S_1,S_2?(!*HS_2*;!*HS_1*),(!*HS_2*;!* HS_3*)* ,0 $\xrightarrow[E]{\emptyset,0}$ 0 *Es gilt:*

Beweis (1) Ist $\mathsf{S}_1 \in E$. Es gilt:

$$\frac{(\mathrm{emit})!\mathsf{HS}_2 \xrightarrow[E]{\{HS_2\},0} 0; \qquad (\mathrm{emit})!\mathsf{HS}_1 \xrightarrow[E]{\{HS_1\},0} 0}{(\mathrm{seq})!\mathsf{HS}_2;!\mathsf{HS}_1 \xrightarrow[E]{\{HS_2,HS_1\},0} 0}$$

$$\frac{(\mathrm{seq})!\mathsf{HS}_2;!\mathsf{HS}_1 \xrightarrow[E]{\{HS_2,HS_1\},0} 0}{(\mathrm{presentcase})\mathsf{S}_1,\mathsf{S}_2?(!\mathsf{HS}_2;!\mathsf{HS}_1),(!\mathsf{HS}_2\ \ ;!\mathsf{HS}_3),0 \xrightarrow[E]{\{HS_2,HS_1\},0} 0}$$

(2) Ist $\mathsf{S}_2 \in E$ $\mathsf{S}_1 \notin E$, dann gilt Es gilt:

$$\frac{(\mathrm{emit})!\mathsf{HS}_2 \xrightarrow[E]{\{HS_2\},0} 0; \qquad (\mathrm{emit})!\mathsf{HS}_3 \xrightarrow[E]{\{HS_3\},0} 0}{(\mathrm{seq})!\mathsf{HS}_2;!\mathsf{HS}_3 \xrightarrow[E]{\{HS_2,HS_3\},0} 0}$$

$$\frac{(\mathrm{seq})!\mathsf{HS}_2;!\mathsf{HS}_3 \xrightarrow[E]{\{HS_2,HS_3\},0} 0}{(\mathrm{presentcase})\mathsf{S}_1,\mathsf{S}_2?(!\mathsf{HS}_2;!\mathsf{HS}_1),(!\mathsf{HS}_2\ \ ;!\mathsf{HS}_3),0 \xrightarrow[E]{\{HS_2,HS_3\},0} 0}$$

(3) Ist S_1, $\mathsf{S}_2 \notin E$, dann gilt Es gilt:

$$\frac{(\mathrm{null})0 \xrightarrow[E]{\emptyset,0} 0}{(\mathrm{presentcase})\mathsf{S}_1,\mathsf{S}_2?(!\mathsf{HS}_2;!\mathsf{HS}_1),(!\mathsf{HS}_2\ \ ;!\mathsf{HS}_3),0 \xrightarrow[E]{\emptyset,0} 0}$$

Lemma 29 (H4) *Seien S_1,S_2,HS_1,HS_2,HS_3 Signale und sei $r := (\{\uparrow$*p*;2 | immediate S_1, immediate $S_2 \nRightarrow$ (!HS_1;!HS_2) (!HS_1;!HS_3)) $\setminus HS_1$, dann gilt:*

(1) $\mathtt{p}$ *terminiert nicht und* $\mathsf{S}_1,\mathsf{S}_2 \notin E_p$. $r \xrightarrow[E]{E_p,0}$ *(*$\{\uparrow \mathtt{p}\text{'};2 \mid$ *immediate* S_1, *immediate* $\mathsf{S}_2 \nRightarrow (!\mathsf{HS}_1;!\mathsf{HS}_2)(!\mathsf{HS}_1;!\ \mathsf{HS}_3\ ))\backslash \mathsf{HS}_1$

(2) $\mathtt{p}$ *terminiert,* $\mathsf{S}_1,\mathsf{S}_2 \in E_p$. *Dann gilt* $r \xrightarrow[E]{E_p*\{HS_2\},0} 0\backslash \mathsf{HS}_1$

(3) $\mathsf{S}_1,\mathsf{S}_2 \in E_p$, *dann gilt* $r \xrightarrow[E]{E_p*\{HS_2\},0} 0\backslash \mathsf{HS}_1$

(4) $\mathsf{S}_1 \in E_p$ *und* $\mathsf{S}_2 \notin E_p$, *dann gilt* $r \xrightarrow[E]{E_p*\{HS_2\},0} 0\backslash \mathsf{HS}_1$

(5) $\mathsf{S}_1 \notin E_p$ *und* $\mathsf{S}_2 \in E_p$, *dann gilt* $r \xrightarrow[E]{E_p*\{HS_3\},0} 0\backslash \mathsf{HS}_1$

Beweis (1) $\mathtt{p}$ terminiert nicht und $\mathsf{S}_1,\mathsf{S}_2 \notin E_p$. Die Behauptung folgt aus dem Lemma 27.

(2) $\mathtt{p}$ terminiert, $\mathsf{S}_1,\mathsf{S}_2 \in E_p$. Die Behauptung folgt aus dem Lemma 27 und Lemma 28 und der Verhaltensregel (signal).

(3) $\mathsf{S}_1,\mathsf{S}_2 \in E_p$. Die Behauptung folgt aus dem Lemma 27 und Lemma 28 und der Verhaltensregel (signal).

(4) $\mathsf{S}_1 \in E_p$ und $\mathsf{S}_2 \notin E_p$. Die Behauptung folgt aus dem Lemma 27 und Lemma 28 und der Verhaltensregel (signal).

(5) $\mathsf{S}_1 \notin E_p$ und $\mathsf{S}_2 \in E_p$. Die Behauptung folgt aus dem Lemma 27 und Lemma 28 und der Verhaltensregel (signal).

Lemma 30 (H5) *Sei* S *ein Signal.*

(1) Ist $\mathsf{S} \in E$, *so folgt* $\mathsf{S}?1^*,0 \xrightarrow[E]{\emptyset,0} 1^*$

(2) Ist $\mathsf{S} \notin E$, *so folgt* $\mathsf{S}?1^*,0 \xrightarrow[E]{\emptyset,0} 0$

Beweis

(1) Ist $\mathsf{S} \in E$, so folgt

$$\dfrac{\dfrac{(\text{unit delay})1 \xrightarrow[E]{\emptyset,1} 0}{(\text{loop})1^* \xrightarrow[E]{\emptyset,1} 0;1^*}}{(\text{present}+)\mathsf{S}?1^*,0 \xrightarrow[E]{\emptyset,1} 0;1^*}$$

Die Behauptung folgt mit Lemma 1.

(2) Ist $\mathsf{S} \notin E$, so folgt $\dfrac{(\text{null})0 \xrightarrow[E]{\emptyset,0} 0}{(\text{present-})\mathsf{S}?1^*,0 \xrightarrow[E]{\emptyset,0} 0}$

Lemma 31 (H6) *Seien S,HE Signale, so folgt*

(1) Ist $S \notin E$*, so folgt* $S?(HE?q,0),0 \xrightarrow[E]{\emptyset,0} 0$

(2) Sind $S,HE \in E$*, so folgt* $S?(HE?q,0),0 \xrightarrow[E]{E_p,k} q'$

(3) Ist $HE \notin E$ *und* $S,HE \in E$*, so folgt* $S?(HE?q,0),0 \xrightarrow[E]{\emptyset,0} 0$

Beweis

(1) Ist $S \notin E$, so folgt $\dfrac{(\text{null})0 \xrightarrow[E]{\emptyset,0} 0}{(\text{present-})S?(HE?q,0),0 \xrightarrow[E]{\emptyset,0} 0}$

(2) Sind $S,HE \in E$, so folgt $\dfrac{\dfrac{q \xrightarrow[E]{E_p,k} q'}{(\text{present+})HE?q,0 \xrightarrow[E]{E_p,k} q'}}{(\text{present+})S?(HE?q,0),0 \xrightarrow[E]{E_p,k} q'}$

(3) Ist $HE \notin E$ und $S,HE \in E$, so folgt $\dfrac{\dfrac{0 \xrightarrow[E]{\emptyset,0} 0}{(\text{present-})HE?q,0 \xrightarrow[E]{\emptyset,0} 0}}{(\text{present+})S?(HE?q,0),0 \xrightarrow[E]{\emptyset,0} 0}$

Lemma 32 *Es gilt die folgende Äquivalenz.*

```
trap ExceptionSignalList in
 p
handle EHE1 do p1
...
handle EHEm do pm
end trap
```

$\Longleftrightarrow$

```
signal traphalt_,
  ExceptionSignalList
in
 signal sc_go_state_23,
 sc_go_state_20 in
  nothing;
  signal sc_weak1 in
   weak abort
    [p]
   when
   case immediate [traphalt_] do
    emit sc_weak1;
    nothing;
    emit sc_go_state_20
   case immediate [ESE1 or ESEn] do
    emit sc_weak1;
    nothing;
    emit sc_go_state_23
   end weak abort;
   present [not sc_weak1]
   then nothing
   end present
  end signal ;
  present [sc_go_state_20]
  then halt
  end present;
  present [sc_go_state_23]
  then
  [
    nothing;
    present [EHE1]
    then nothing;[p1]
    else nothing
    end present
   || ... ||
    nothing;
    present [EHEm]
    then nothing;[pm]
    else nothing
    end present
   ]
  end present
 end signal
end signal
```

BEWEIS Mit Hilfe der oben aufgestellten Lemmata folgt das beabsichtigte Verhalten des SyncCharts. Da $\mathsf{w1}?0{,}0 \xrightarrow[E]{\emptyset,0} 0$ gilt,wird der Ausdruck ignoriert.

Wird in p eine *Trap* mit höherer Priorität durch ein `exit` ausgelöst, so sendet die Umwandlung von `exit` das th. Somit folgt aus Lemma 29 und Lemma 30 der schwache Abbruch von p und der Übergang in einen Zustand ewiger Pause, der nur durch einen Abbruch verlassen werden kann.

$$\frac{(\mathrm{H4})(\{\uparrow \mathsf{p};2 \mid \text{immediate th, immediate ES} \nRightarrow (\mathsf{w1};0;\mathsf{go20}),(\mathsf{w1};0;\mathsf{go23})\}\ )\backslash \mathsf{w1} \xrightarrow[E]{E_p * go20,0} 0 \qquad (\mathrm{H5})\mathsf{go20}?1^*, 0) \xrightarrow[E]{E_p * go20,0} 1^*}{(\{\uparrow \mathsf{p};2 \mid \text{immediate th, immediate ES} \nRightarrow (\mathsf{w1};0;\mathsf{go20}),(\mathsf{w1};0;\mathsf{go23})\}\ )\backslash \mathsf{w1};\mathsf{go20}?1^*, 0) \xrightarrow[E]{E_p * go20,0} 1^*}$$

Wird keine übergeordnete *Trap*, sondern diese *Trap* durch p ausgelöst, so folgt ebenfalls aus Lemma 29 der schwache Abbruch von p.

$$\frac{(\mathrm{H4})(\{\uparrow \mathsf{p};2 \mid \text{immediate th, immediate ES} \nRightarrow (\mathsf{w1};0;\mathsf{go20}),(\mathsf{w1};0;\mathsf{go23})\}\ )\backslash \mathsf{w1} \xrightarrow[E]{E_p * go23,0} 0}{(\{\uparrow \mathsf{p};2 \mid \text{immediate th, immediate ES} \nRightarrow (\mathsf{w1};0;\mathsf{go20}),(\mathsf{w1};0;\mathsf{go23})\}\ )\backslash \mathsf{w1};\mathsf{go20}?1^*, 0) \xrightarrow[E]{E_p * go23,0} 1^*}$$

Aufgrund der Sequenz folgt nun die Ausführung aller ausgelösten *Handler* p, wobei für $\mathsf{EHE}i?0;\mathsf{p}_i,0) \xrightarrow[E]{\emptyset,0} 0$ gilt falls $\mathsf{EHE}i \notin E$ und $\mathsf{EHE}i?0;\mathsf{p}_i,0) \xrightarrow[E]{E_{pi},k} p_i$ falls $\mathsf{EHE}i \in E$

$$\frac{(0;\mathsf{EHE1}?0;\mathsf{p}_1,0),0) \mid \ldots \mid ((0;\mathsf{EHEm}?0;\mathsf{p}_m,0),0) \xrightarrow[E]{E_{p1} * E_{pm},k} p_1 \mid \ldots \mid p_m.}{\mathsf{go23}?((0;\mathsf{EHE1}?0;\mathsf{p}_1,0),0) \mid \ldots \mid ((0;\mathsf{EHEm}?0;\mathsf{p}_m,0),0) \xrightarrow[E]{E_{p1} * E_{pm},k} p_1 \mid \ldots \mid p_m}$$

Sind alle p_i terminiert so folgt $(0 \mid \ldots \mid 0) \xrightarrow[E]{E_{p1} * E_{pm},0} 0$ und die vorhanden lokalen Signale werden aus der Umgebung entfernt.

Tritt keiner der beiden Fälle ein, so folgt dass p entweder in p' übergeht, wodurch in der nächsten Instanz wieder alle Fälle zu betrachten sind oder p terminiert, dann folgt aus Lemma 29 und Lemma 31 die Terminierung inklusive des entfernens der lokalen Signale.

3.4.25. Der Befehl exit

Esterelgrammatik

⟨*Exit*⟩ ::= 'exit' <ExceptionIdentifier> ['('<Expression>')']

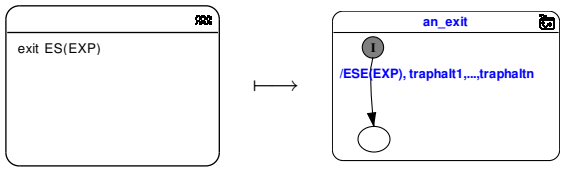
Beschreibung des Befehls

Der `exit`-Befehl emitiert ein Ausnahmesignal, welches eine `trap` auslöst. Das Ausnahmesignal verhält sich wie ein normales Signal. Eventuell vorhandene Sequenzen, die in der Hierachie zwischen dem `exit` und der zugehörigen `trap` liegen, halten vor dem nächsten Befehl an.

Beschreibung eines äquivalenten Makrozustandes

Der `exit`-Befehl wird ähnlich wie der `emit` Befehl 3.4.11 umgewandelt. Dabei wird statt des Ausnahmesignals ein, in der zum Ausnahmesignal gehörenden Umwandlung, deklariertes Signal gesendet. Zusätzlich werden die traphalt-Signale, der in der Hierachie zwischen dem `exit` und der zugehörigen `trap` liegenden, umgewandelten *Traps* gesendet. Dadurch werden nur diese *Traps* angehalten. *Traps* die zwar der ausgelösten *Trap* untergeordnet sind, aber parallel zum `exit` ausgeführt werden, werden so nicht beeinflußt.

Transformationsregel 27 (exit)



Das Auslösen einer Ausnahme mit dem entsprechenden Verhalten muß im Zusammenhang mit der Umwandlung der zugehörigen *Trap* betrachtet werden.

Lemma 33 *Es gilt die folgende Äquivalenz.*

```
exit ESE(EXP)   ⟺   emit ESE;
                    emit traphalt1;
                    ...
                    emit traphaltn;
                    halt
```

Beweis Es gilt aufgrund von mehreren Anwendungen von (emit).

$!\mathsf{ESE};!\mathsf{traphalt}_1;\ldots;!\mathsf{traphalt}_n;1^* \xrightarrow[E]{\{ESE\}*\{traphalt_1\}*\ldots*\{traphalt_n\},0} 0;1^*.$

Durch 1^* hält die weitere Ausführung an, es werden also insbesondere keine Sequenzen mehr fortgesetzt. Durch die *traphalt* Signale werden alle *Traps* mit geringere

Priorität als die Ausgelöste angehalten. Somit ist das Verhalten der Umwandlung äquivalent `exit`.

3.4.26. Der Befehl weakabort

Esterelgrammatik

⟨ *WeakAbort* ⟩ ::= 'weak abort' <Statement> 'when' (<DelayExpression> ['do' <Statement>] end' ['weak abort']
| 'weak abort' <Statement> 'when' <AbortCase> <AbortCase> 'end' ['weak abort']

Beschreibung des Befehls

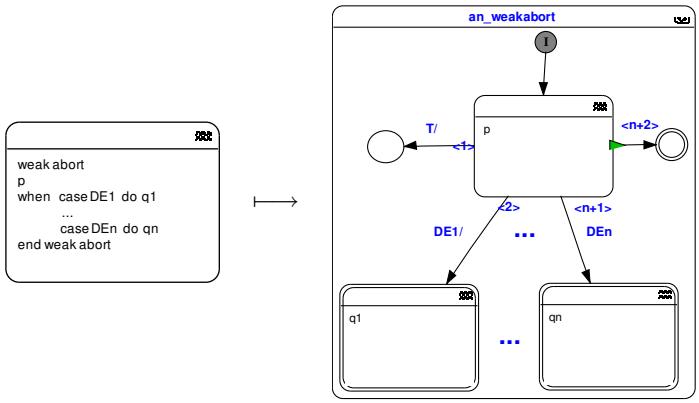
Der `weak abort` Befehl bricht die Ausführung eines <Statement> am Ende der aktuellen Instanz ab, sobald eine <DelayExpression> erfüllt ist. Tritt keine <DelayExpression> auf und terminiert das syntax<Statement>, so terminiert auch der `weak abort` Befehl. Existiert zu der ausgelösten <DelayExpression> ein `do` Teil, so wird dieser Teil in der gleichen Instanz nach Abbruch des <Statement> ausgeführt. Werden mehrere Fälle durch das Schlüsselwort *case* beschrieben, so entspricht die textliche Reihenfolge der Priorität, wobei der textlich erste Fall, der mit der höchsten Prioritäten ist.

Beschreibung eines äquivalenten Makrozustandes

Man kann ein `weak abort` nachstellen, indem man vom Makrozustand des Umgewandelten <Statement>, welches abbgeborchen werden kann, für jeden angegebenen Fall eine *weak abortion* die zu einem Makrozustand führt, darstellt. Der Makrozustand, der das Ziel einer *weak abortion* ist, entspricht der Umwandlung des entsprechenden <Statement> oder der Umwandlung von `nothing` falls kein `do` <Statement> vorhanden ist. Die Prioritäten der *weak abortion* entsprechend der textlichen Reihenfolge der Fälle.

Ist der `weak abort` Befehl einer *Trap* untergeordnet, so wird zusätzlich eine *weak abortion* mit höchster Priorität, die zu einem einfachen Zustand ohne ausgehende Transitionen führt, hinzugefügt. Diese *weak abortion* wird durch ein # T augelöst. Dabei ist das T eine Oder-Verknüpfung vom traphalt 3.4.24 Signal und den lokales Signalen, welcher der Umwandlung eines Ausnahmesignales entsprechen, der in der Hierarchie nächst höheren *Trap*. Wird aufgrund, einer Ausführung eines Transferierten `exit` T wahr, so wird insbesondere `p` am Ende der Instanz abgebrochen, aber die Umwandlung von `werk abort` wird so in der weiteren Ausführung unterbrochen. Auf diese Weise wird verhindert, dass die Umwandlung des `weak abort`-Befehl Vorrang vor einer *Trap* besitzt.

Transformationsregel 28 (weakabort)



Transformationsregel 29 (weak abort in weak abort case)
weak abort p when DE ⟼ *weak abort p when case DE do nothing*

3.5. Nicht transformierbare Befehle

Es ist nur der `exec` nicht in ein SyncCharttransformierbar.

Esterelgrammatik von `exec`

⟨*Exec*⟩ ::= ‘exec’ <TaskIdentifier>
‘(’ { <VariableIdentifier> } >‘)’
‘(’{ <Expression>}‘)’
‘return’ <SignalIdentifier> [‘do’ <Statement>
‘end’ [‘task’]]
| ‘exec’ ‘case’ <TaskIdentifier>
‘(’ { <VariableIdentifier> } ‘)’
‘(’{ <Expression> }‘)’
‘return’ <SignalIdentifier> [‘do’ <Statement>
{ ‘case’ <TaskIdentifier>
‘(’ { <VariableIdentifier> } >‘)’
‘(’{ <Expression>}‘)’
‘return’ <SignalIdentifier> [‘do’ <Statement>]
}
‘end’ [‘task’]

Beschreibung des Befehls

Der `exec`-Befehl ruft einen Task auf, welcher extern definiert wird. Dieser Aufruf signalisiert seine Beendigung durch emitieren des ihm exklusiv zugeordneten Signals

⟨*SignalIdentifier*⟩. Dabei können mehrere Variablen übergeben werden. Die Werte dieser Variablen können durch den Task verändert werden. Zusätzlich können Ausdrücke übergeben werden. Die Reihenfolge der Typen der Parameter muss mit der in der Deklaration zu Beginn des Moduls identisch sein.

Beschreibung eines äquivalenten Makrozustandes

Es ist keine Umwandlung möglich, da SyncCharts keinen Aufruf von Task zulassen, es sei denn es wird ein textlicher Makrozustand benutzt, in den der Esterel-Programmcode geschrieben wird.

4. Optimierung von SyncCharts

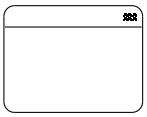
In diesem Kapitel werden die verschieden Regeln vorgestellt, um in einem korrekten SyncChart unnötige Zustände und Transitionen zu entfernen. Dabei handelt es sich nur um eine strukturelle Optimierung. Es wird nicht das Verhalten analysiert und optimiert, sondern nur der Aufbau des SyncCharts, um insbesonders die überflüssigen Komponenten, die durch eine Umwandlung 3 von Esterel in ein SyncChart enstehen, zu entfernen. Diese Optimierung ist aber für jedes SyncChart möglich.

Die folgenden Regeln betrachten jeweils nur einen Makrozustand, wobei kein Zustand interne Aktionen besitzen darf. Zu diesen Regeln wird jeweils eine linke Seite und die zugehörige rechte Seite angegeben, die die Veränderungen darstellt.

Es werden die folgenden grafischen Objekte als Abkürzung verwendet.

: Der Makrozustand mit dem Namen *grey cloud* stellt eine beliebige Umgebung von Transitionen und Zuständen eines SyncChart dar.

: Diese Transition steht für beliebige zulässige Transitionen.



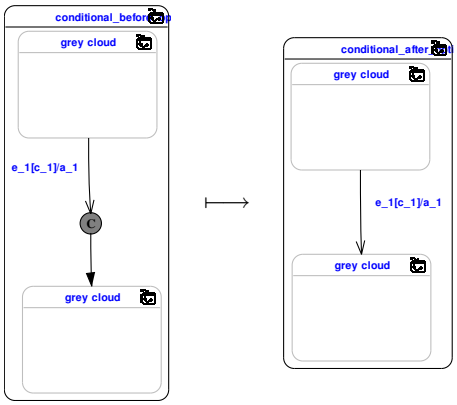
: Dieser Zustand steht für jede Art von Zuständen.

In den Regeln sind die Transitionsbeschriftungenin der Form $e[c]/a$ angegeben, wobei Indizierungen vorkommen. Dabei steht e für einen beliebigen *Trigger*, c für eine beliebige Kondition und a für beliebige Aktionen. Am Ende des Kapitels wird ein Algorithmus 4.1 angegeben, der einen Makrozustand optimiert. Die folgenden Regeln sind entsprechend ihres Auftretens im obengenannten Algorithmus angegeben.

4.0.1. Entfernen unnötiger *Conditional*-Pseudozustände

Diese Regel entfernt alle *Conditional*-Pseudozustände, die nur eine ausgehende Transitionohne Beschriftung besitzen. Dazu werden die Ziele aller eingehenden Transitionen des *Conditional*-Pseudozustandes auf das Ziel der ausgehenden Transition des *Conditional* punktiert. Anschließend wird der Zustand sowie die ausgehende Transition aus dem Statechart entfernt.

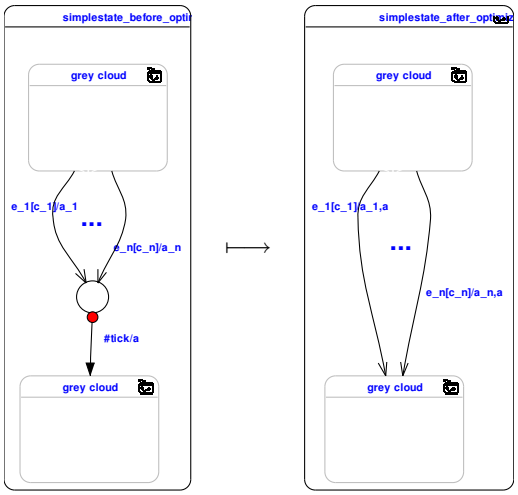
Optimierungsregel 1 (Entfernen unnötiger *Conditional*-Pseudozustände)



4.0.2. Entfernen unnötiger, einfacher Zustände

Zunächst werden die einfachen Zustände entfernt, die nur eine ausgehende Transition, mit $e = \#\ \mathsf{tick}$ und $c = \emptyset$, besitzen. Alle eingehenden Transitionen werden dann auf das Ziel der ausgehenden Transition umgeleitet. Die Aktionen der ausgehenden Transition werdern den Aktionen der umgeleiteten Transitionen jeweils hinzugefügt. Außerdem wird der einfache Zustand und die ausgehende Transition aus den SyncChart entfernt.

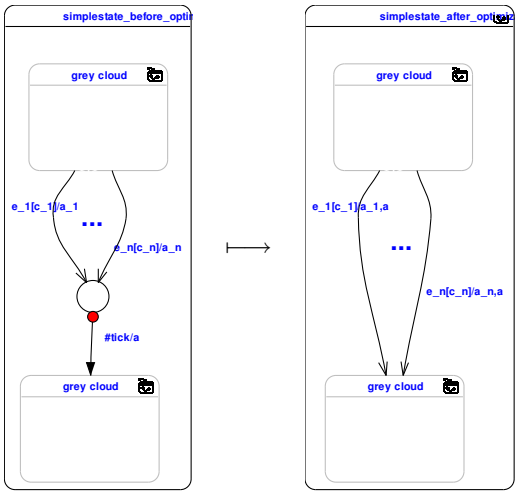
Optimierungsregel 2 (Entfernen unnötiger, einfacher Zustände 1)



Eine weitere Regel zum Entfernen einfacher Zustände wird angewendet, sobald die eingehenden Transition und die ausgehende Transition den gleichen Trigger, der nicht *immediate* sein darf, besitzen. In diesem Fall werden die in einer Transitionsbeschriftung 2.3.2 angegebenen $\langle factor \rangle$ addiert, und als Summe der eingehenden

Transitionen gesetzt. Diese wird auf das Ziel der ausgehenden Transitiongerichtet. Der Zustand und die ausgehende Transition werden entfernt.

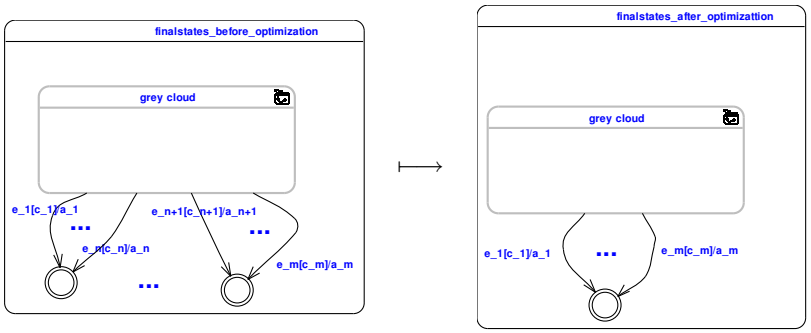
Optimierungsregel 3 (Entfernen unnötiger, einfacher Zustände 2)



4.0.3. Zusammenfassen einfacher, finaler Zustände

Sind in einen Makrozustand mehrere einfache, finale Zustände vorhanden, so werden alle eingehenden Transitionen der einfachen, finalen Zustände auf einen einfachen, finalen Zustand gerichtet und alle anderen einfachen, finalen Zustände aus dem Sync-Chart entfernt.

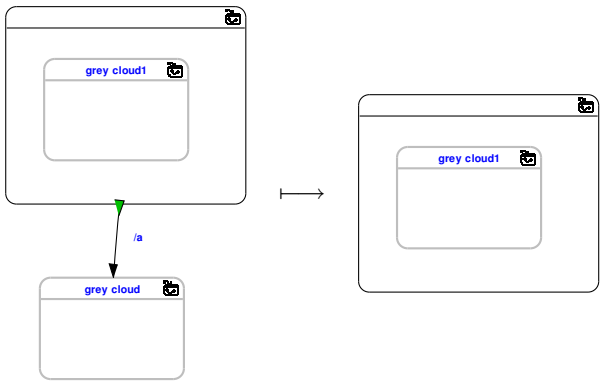
Optimierungsregel 4 (Zusammenfassen einfacher, finaler Zustände)



4.0.4. Entfernen unnötiger *normal terminations*

Besitzt ein Makrozustand keine finalen Zustände, so wird eine vorhanden ausgehende Transition aus dem SyncChart gelöscht, da sie niemals ausgelöst wird. Entstehen dadurch Zustände, die keine eingehende Transition mehr besitzen, so werden diese ebenfalls aus dem SyncChart entfernt. In der Regeldarstellung gelte, dass in *grey cloud1* auf der obersten Hierarchieebene keine finalen Zustände vorhanden sind.

Optimierungsregel 5 (Entfernen unnötiger *normal terminations*)



4.0.5. Entfernen unnötiger Makrozustände

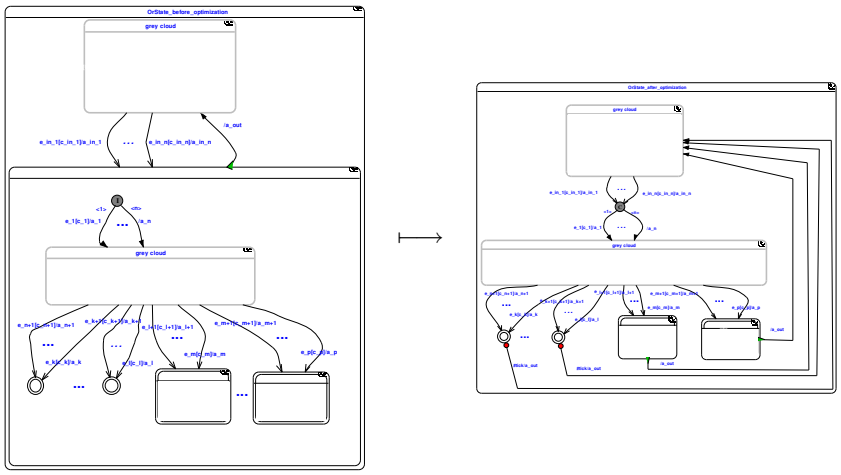
Um einen Makrozustand zu entfernen, müssen die folgenden Bedingungen erfüllt sein:

- Der Makrozustand darf keine ausgehende *weak abortion* oder *strong abortion* besitzen,
- der Makrozustand darf keine lokalen Signale oder Variablen deklarieren,
- es darf kein paralleler Makrozustand sein,
- es existiert ein übergeordneter Makrozustand.

Sind diese Bedingungen erfüllt, wird der Initiale Zustand in einen *Conditional*-Pseudozustand umgewandelt. Alle beim dem Makrozustand eingehenden Transitionen werden auf den *Conditional*-Pseudozustand gelenkt. Dadurch bleibt das Verhalten zu Beginn des Makrozustandes gleich. Alle finalen Makrozustände werden in die entsprechenden nicht-finalen Makrozustände umgewandelt und erhalten als ausgehende Transition eine Kopie der eventuell vorhandenen *normal termination*. Alle finalen, einfachen Zustände werden in einfache Zustände umgewandelt und erhalten, falls eine *normal termination* vorhanden ist, eine *strong abortion*,welche zum Ziel der

normal termination führt. Dabei wird $e := \#$ *tick* gesetzt, um einen sofortigen Abbruch des einfachen Zustandes herbeizuführen. Die Aktion wird von der ausgehenden Transition übernommen. Diese einfachen Zustände werden später mit der Regel 4.0.2 entfernt. Somit ist auch ein äquivalentes Verhalten bei der Terminierung des Makrozustandes sichergesellt. Der Makrozustand mit seiner ausgehenden Transition kann wird aus dem SyncChart entfernt und sein Inhalt wird dem übergeordneten Makrozustand zugewiesen.

Optimierungsregel 6 (Entfernen unnötiger Makrozustände)



4.0.6. Entfernen von Makrozuständen mit nur zwei Unterzuständen

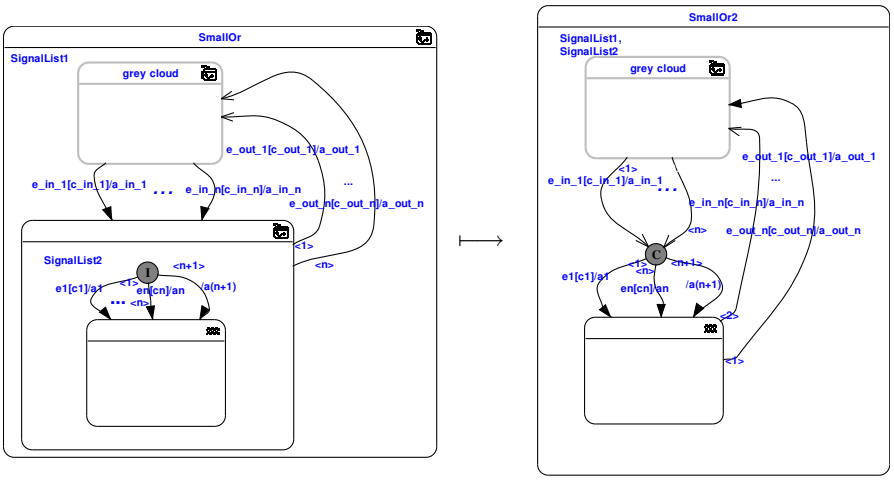
Ein Spezialfall sind Makrozustände, die nur zwei Unterzustände besitzen, wovon nur einer kein Pseudozustand also ein beliebiger Zustand ist. Diese können problemlos entfernt werden, da sie primär der Weiterleitung der Ausführung dienen. Es darf sich jedoch auch nicht um einen parallelen Makrozustand handeln und es muß ein übergeordneter Makrozustand existieren.

Der Vorgang verläuft analog zu dem in der Regel 4.0.5 beschriebenen Vorgehen. Zusätzlich werden eventuelle ausgehende *weak abortion*s und *strong abortion*s zu den ausgehenden Transitionen des Zustandes hinzugefügt und die lokalen Deklarationen dem übergeordneten Makrozustand hinzugefügt.

Optimierungsregel 7 (Entfernen von Makrozuständen mit nur zwei Unterzuständen)

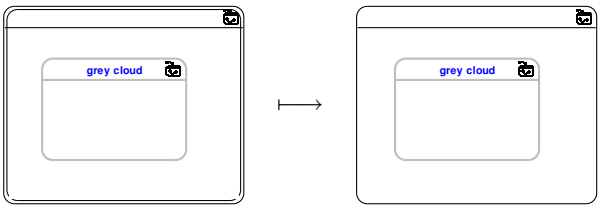
4.0.7. Überprüfen auf finalen Charakter

Besitzt ein finaler Makrozustand, keine untergeordneten finalen Zustände, so wird der Makrozustand in einen entsprechenden nicht-finalen Zustand umgewandelt. Eventu-



ell vorhandene *normal termination* werden zuvor mittels der Regel 4.0.4 entfernt. Es wird in der Regeldarstellung davon ausgegangen, dass in der obersten Hierarchieebene von *grey cloud* keine finalen Zustände existieren.

Optimierungsregel 8 (Überprüfen auf finalen Charakter)



4.1. Der Algorithmus zum Optimieren eines Makrozustandes

Nachdem nun alle Regeln zur Optimierung eines SyncCharts angegeben wurden, wird nun die Anwendung der Regeln, insbesondere die Reihenfolge der Anwendungen, beschrieben.

M sei dazu ein Makrozustand. Eine Regel wird nur ausgeführt, wenn die Bedingungen für ihre Anwendung erfüllt sind.

1. Wende die Regel 4.0.1 auf alle *Conditional*-Pseudozustände aus M an.
2. Wende die Regel 4.0.2 auf alle einfachen Zustände aus M an.
3. Wende die Regel 4.0.3 auf M an.

4. Wende die Regel 4.0.4 auf M an.
5. Wende die Regel 4.0.5 auf M an.
6. Falls M noch im Statechart vorhanden ist, wende die Regel 4.0.6 auf M an.
7. Falls M noch im Statechart vorhanden ist, wende die Regel 4.0.7 auf M an.

Nach der Anwendung dieses Algorithmus sollte sich das Verhalten des SyncChart nicht verändert haben. Der Beweis zur Korrektheit dieser Optimierung eines SyncCharts wird in dieser Arbeit nicht erbracht. Der Algorithmus ist in *KIEL* implementiert worden und liefert korrekte Ergebnisse.

5. Transformation am Beipiel von *ABRO*

In diesem Kapitel werden die Ergebnisse aus Kapitel 3 und Kapitel 4 am Beispiel von *ABRO* vorgestellt. Dieses Esterel Programm wird in den meisten Arbeiten zum Thema Esterel oder Statecharts verwendet und erschient auch hier angemessen. Dabei wird zunächst aus dem Esterel Programmcode mit Hilfe der Regeln aus Kapitel 3 ein SyncChart erzeugt und dieses anschließend mit den Regeln aus dem Kapitel 4 optimiert. Der Anfang und das Ende dieses Beispiels sind bereits in der Abbildung 1.2 angegeben.

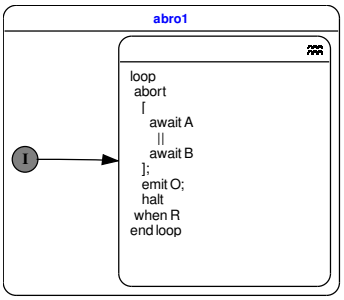
An dieser Stelle wird das Verhalten von *ABRO* vorgestellt. Zunächst wird auf die Signale A und B gewartet und anschließend das Signal O gesendet. Wird das Signal R empfangen, so wird der Vorgang abgebrochen und neu gestartet.

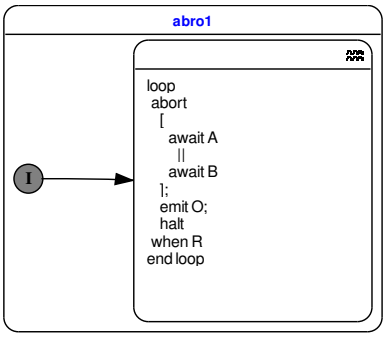
5.1. Beispielhafte Transformation von Esterel nach SyncChart

Das Programm *ABRO* wird folgendermaßen in ein SyncChart transformiert. Zuerst wird das `module` gemäß der Transformationsregel 1 umgewandelt. Dadurch werden insbesondere die Deklarationen in das SyncChart übernommen. Als nächstes wird `loop p end loop` mittels der Transformationsregel 17 umgewandelt. Die Transformationsregel 5 wandelt nun das Kommando `abort p when R` um und erzeugt, da keine optionalen Befehle angegeben wurden, zwei einfache, finale Zustände, wobei einer über eine *normal termination* und der andere über eine *strong abortion* erreicht wird. Darauf folgend wird mit der Transformationsregel 23 die Sequenz `[await A || await B]; emit O; halt` entfernt. Nun werden die parallen Befehle `await A` und `await B` durch die Transformationsregel 19 und die Transformationsregel 8 umgewandelt. Da bei den `await`-Befehlen kein optionaler Befehl für das Verhalten nachdem eintreten der Signale A,B angegeben ist, werden einfache, finale Zustände eingesetzt. Nun folgt die Transformation von `emit O` durch Transformationsregel 12 und zum Abschluß die Umwandlung von `halt` durch die Transformationsregel 3.

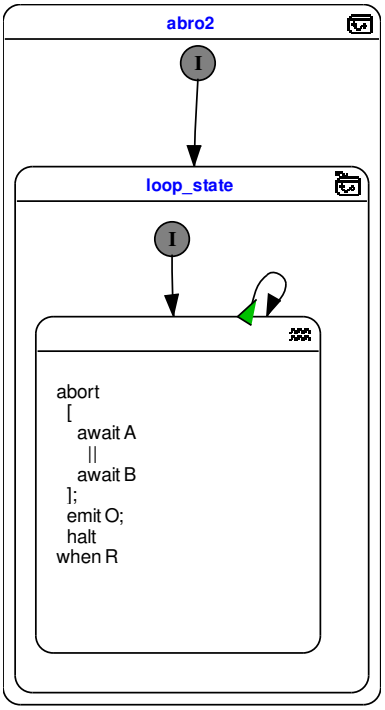
```
module abro:
input A,B,R;
output O;
loop
 abort
   [ await A
     ||
     await B];
  emit O;
  halt
 when R
end loop
end module
```

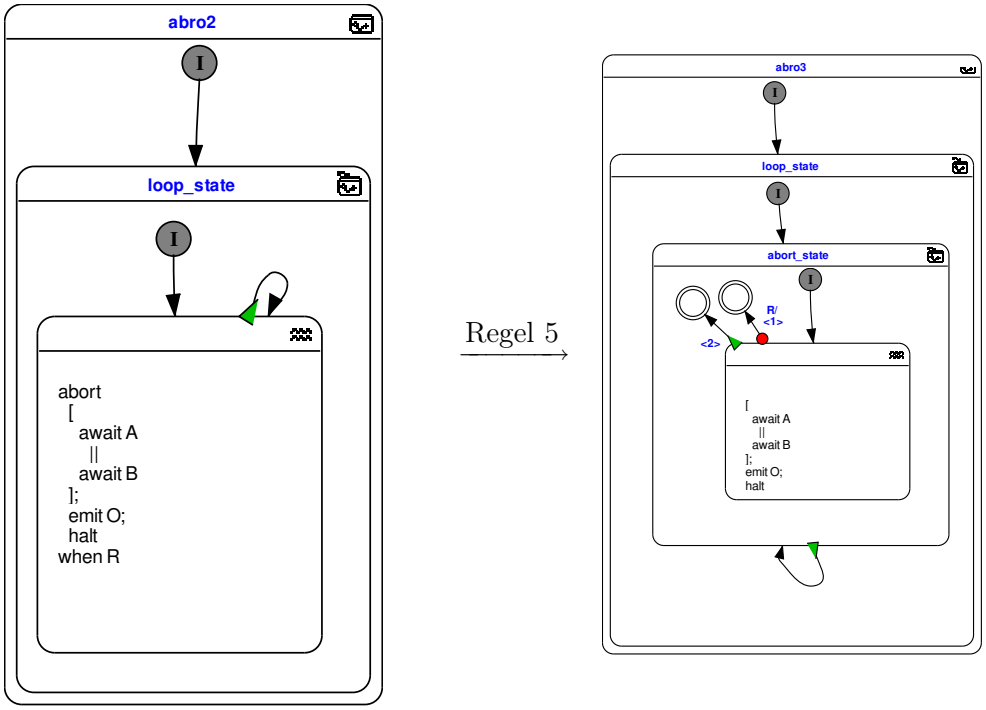
Regel 1 ⟶

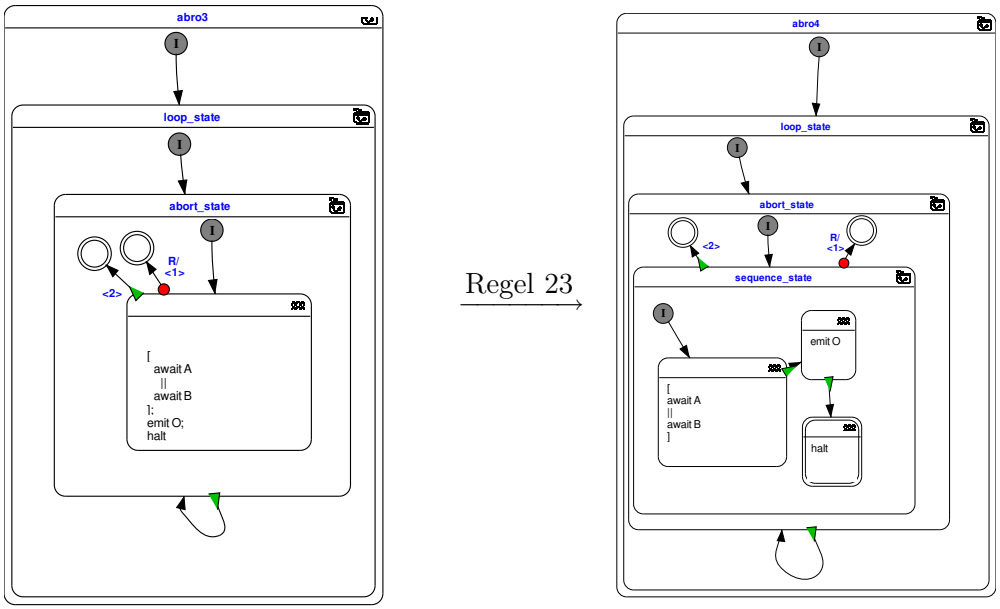


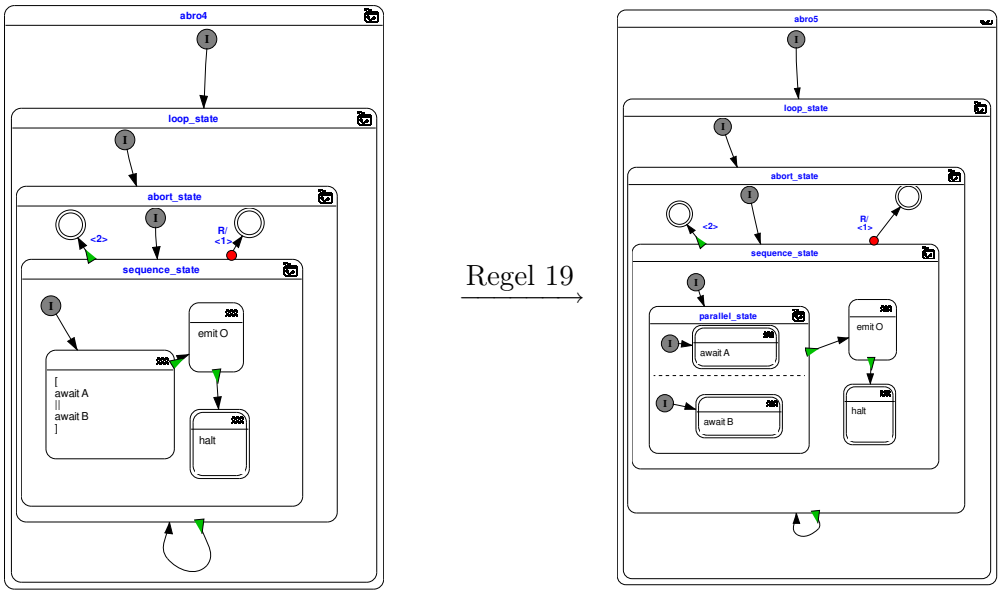


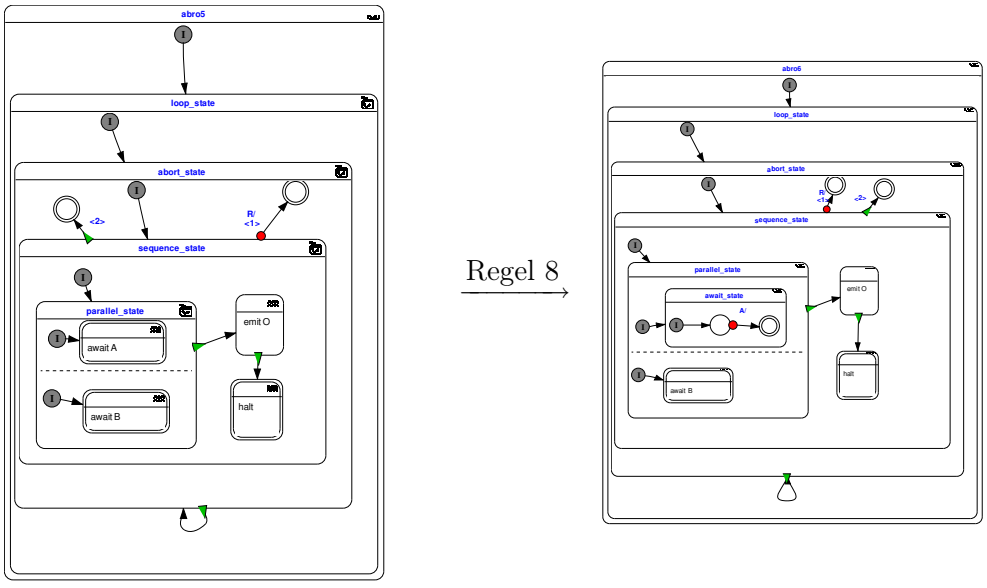
Regel 17 ⟶

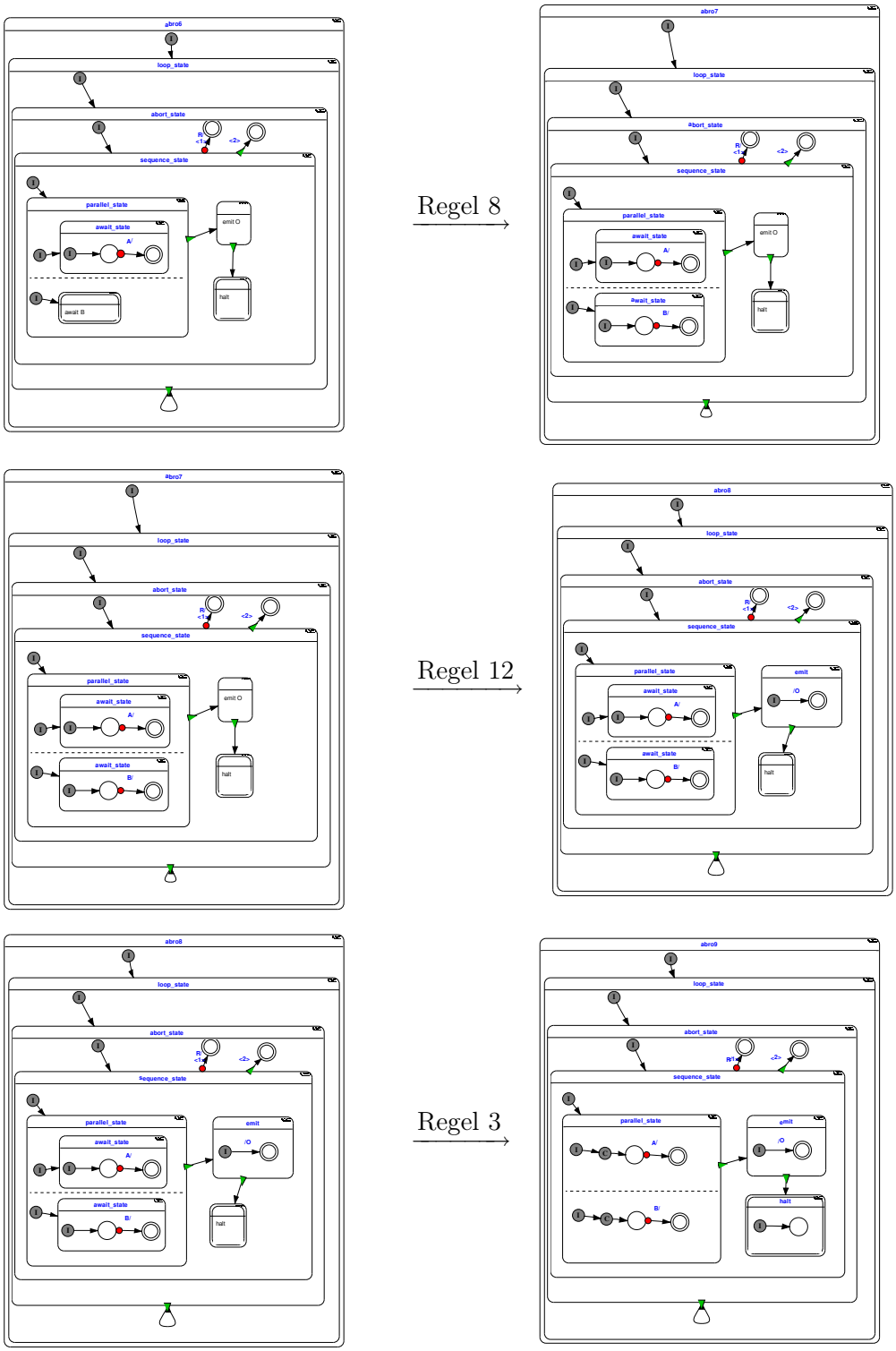



abro2
loop_state
abort
[
await A
||
await B
];
emit O;
halt
when R
Regel 5
abro3
loop_state
abort_state
R/
<1>
<2>
[
await A
||
await B
];
emit O;
halt


abro3
loop_state
abort_state
R/
<1>
<2>
[
await A
||
await B
];
emit O;
halt
Regel 23
abro4
loop_state
abort_state
<2>
R/
<1>
sequence_state
emit O
[
await A
||
await B
]
halt


abro4
loop_state
abort_state
<2>
R/
<1>
sequence_state
emit O
[
await A
||
await B
]
halt
Regel 19
abro5
loop_state
abort_state
<2>
R/
<1>
sequence_state
parallel_state
emit O
await A
await B
halt

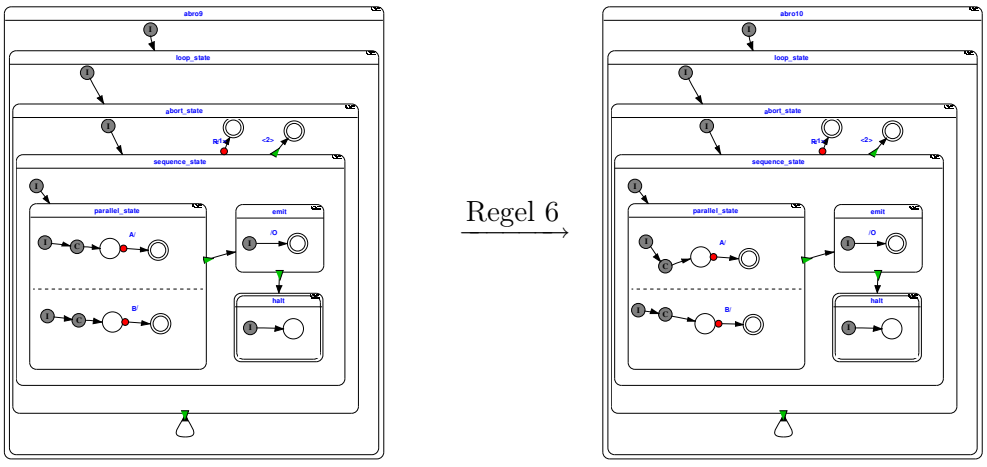

abro5
loop_state
abort_state
<2>
R/
<1>
sequence_state
parallel_state
emit O
await A
await B
halt
Regel 8
abro6
loop_state
abort_state
sequence_state
parallel_state
await_state
A/
emit O
await B
halt

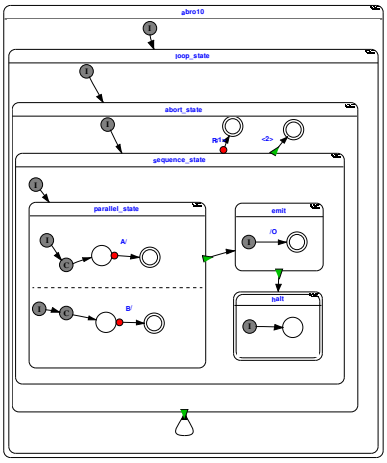


Es ist nun ein Statechart aus dem Esterel Programm entstanden, welches als nächstes Optimiert wird.

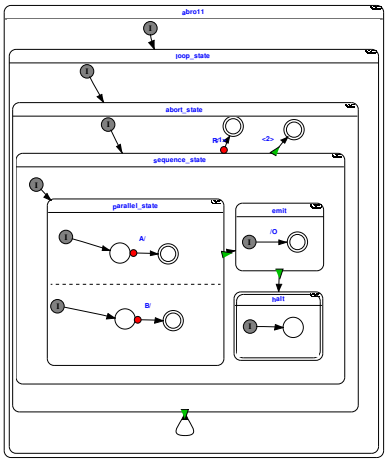
5.2. Optimierung

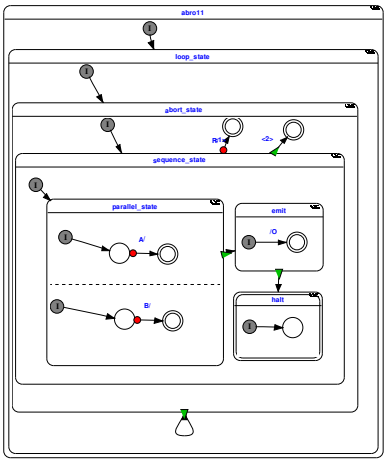
Zunächst werden die beiden parallen Zustände mittels der Regel 6 nacheinander aufgelöst. Anschließend werden die gerade enstandenen *Conditional*-Pseudozustände durch Regel 1 aus dem SyncChartentfernt, und es werden die Zustände, welche aus `emit O` und `halt` hervorgegangen sind, durch die Regel 6 aufgelöst. Wieder werden die *Conditional*-Pseudozustände durch Regel 1 aus dem SyncChart entfernt. Nun wird durch die Regel 2 die *strong abortion* mit der Transitionsbeschriftung `# tick` und der Zielzustand entfernt, da dieser nur durch die *strong abortion* erreichbar war. Aufgrund der Regel 5 wird die *normal termination* entfernt. Der Zielzustand wird ebenfalls aus dem SyncChart gelöscht, da er nicht mehr erreichbar ist. Der Zustand mit dem Namen *abort_state* wird durch die Regel 6 aufgelöst. Im nächsten Schritt wird der *Conditional*-Pseudozustand durch die Regel 1 aus dem SyncChart entfernt, da er nur die *Default*-Transition besaß. Der einfache Zustand, an dem die *strong abortion* mit der Transitionsbeschriftung `# tick` wird nun entfernt und die *strong abortion* mit der Transitionsbeschriftung `R` wird auf den Makrozustand *sequence_state* gelenkt. Als letztes wird der Makrozustand *loop_state* mittels der Regel 6 aufgelöst und der enstandene *Conditional*-Pseudozustand durch die Regel 1 aus dem SyncChart entfernt. Das Ergebnis ist die optimale Darstellung des Esterel Programms *ABRO* als SyncChart.



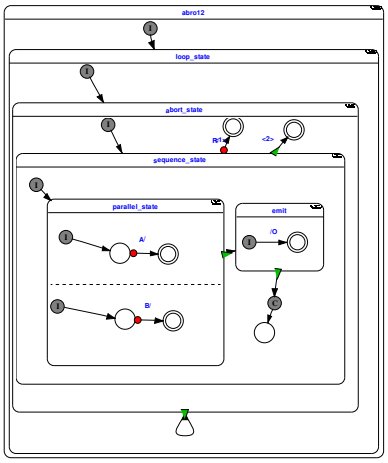


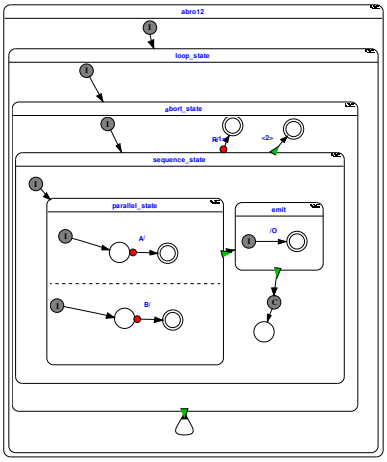
Regel 1



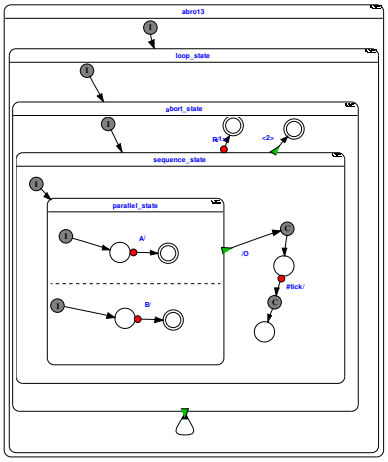


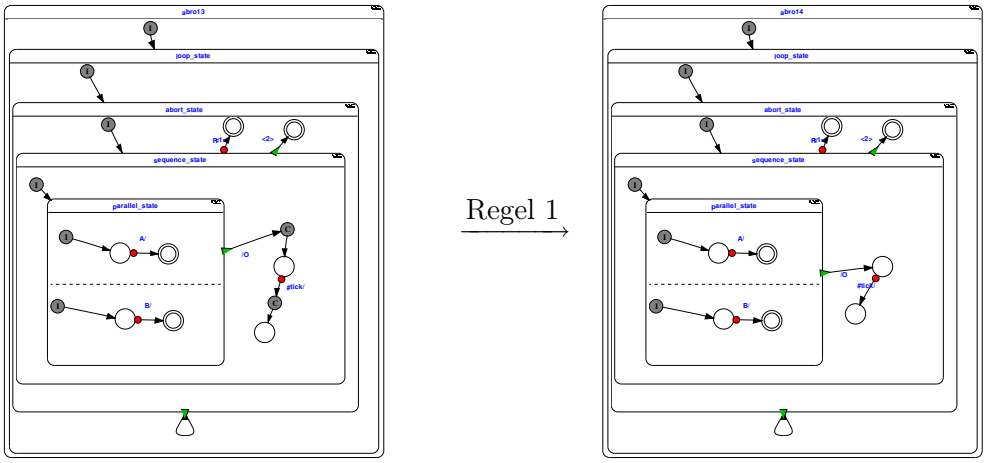
Regel 6

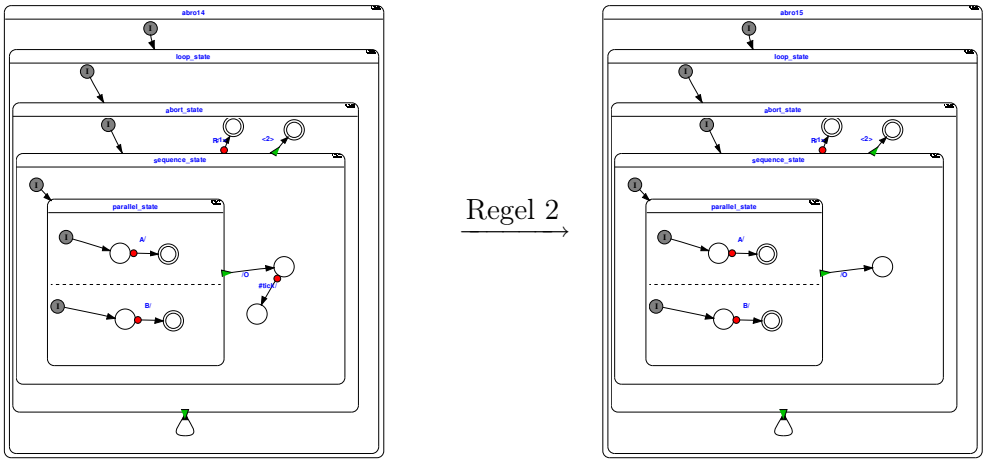


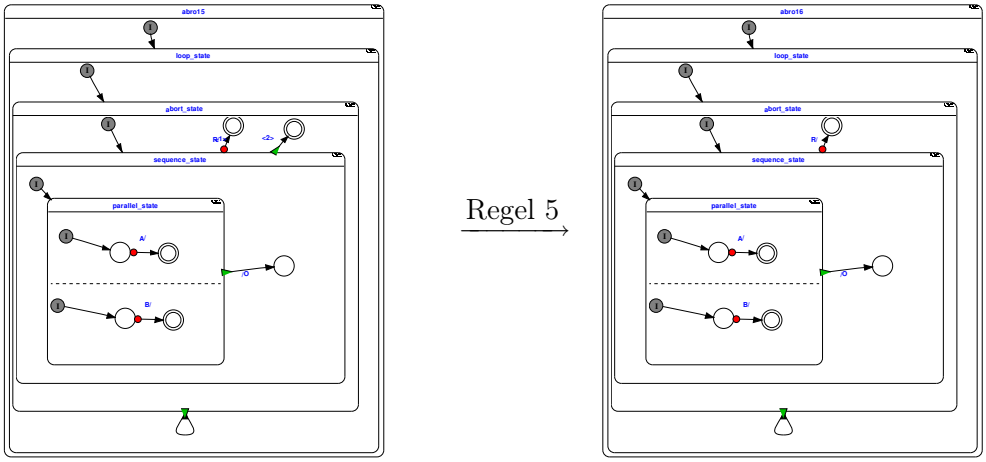


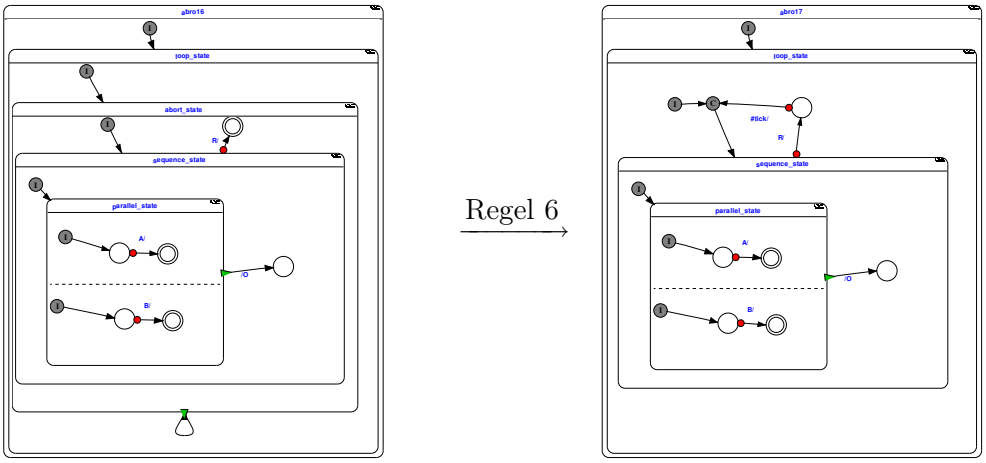
Regel 6

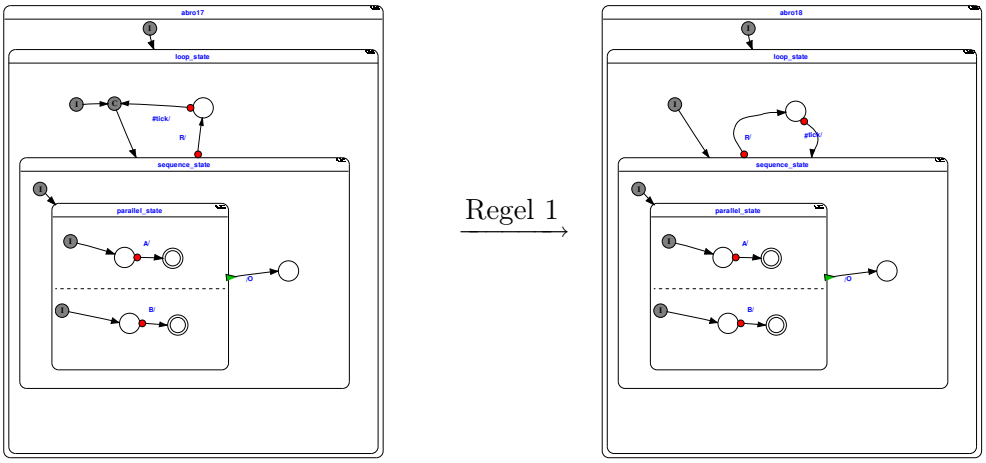


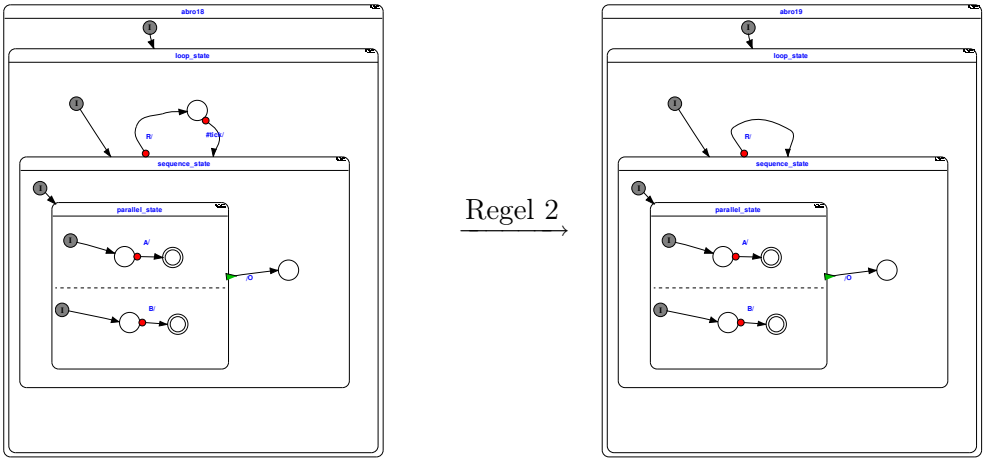

abro13
loop_state
abort_state
R/
<2>
sequence_state
parallel_state
A/
/O
B/
#tick/
abro14
Regel 1

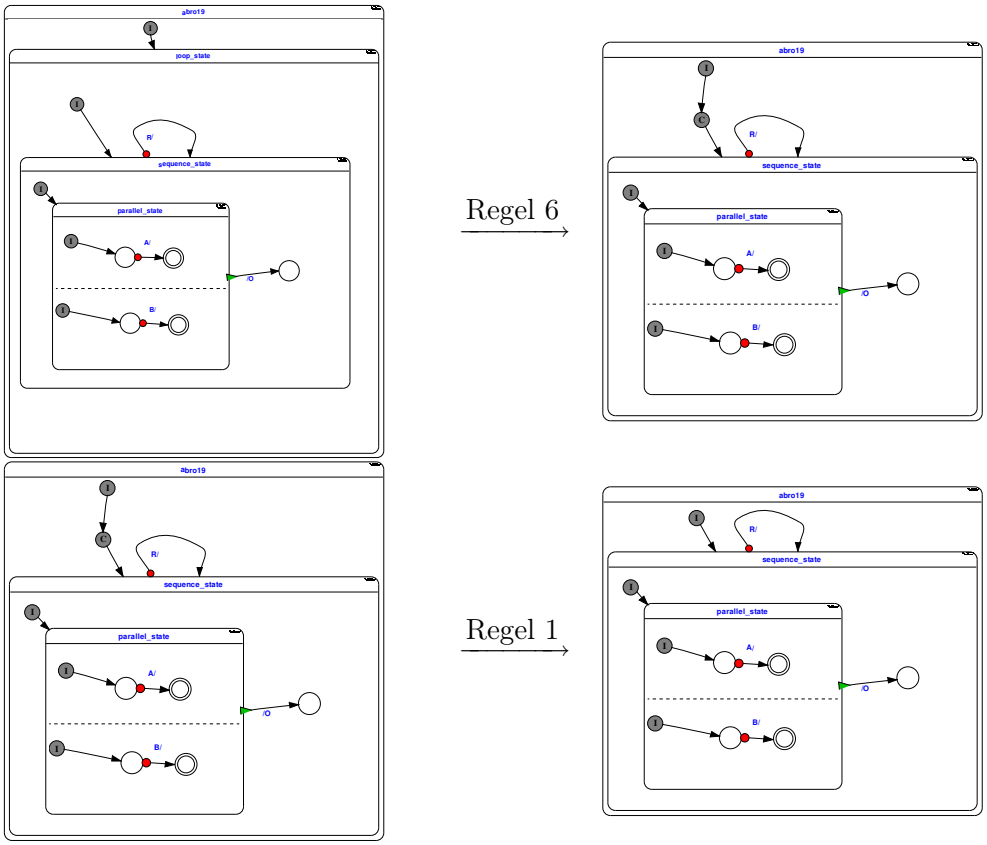

abro14
loop_state
abort_state
R/
<2>
sequence_state
parallel_state
A/
/O
B/
#tick/
abro15
Regel 2


abro15
loop_state
abort_state
R/
<2>
sequence_state
parallel_state
A/
/O
B/
abro16
Regel 5


abro16
loop_state
abort_state
R/
sequence_state
parallel_state
A/
/O
B/
Regel 6
abro17
loop_state
#tick/
R/
sequence_state
parallel_state
A/
/O
B/


abro17
loop_state
#tick/
R/
sequence_state
parallel_state
A/
/O
B/
Regel 1
abro18
loop_state
R/
#tick/
sequence_state
parallel_state
A/
/O
B/


abro18
loop_state
R/
#tick/
sequence_state
parallel_state
A/
/O
B/
Regel 2
abro19
loop_state
R/
sequence_state
parallel_state
A/
/O
B/


abro19
loop_state
sequence_state
parallel_state
A/
/O
B/
R/
Regel 6
Regel 1

6. Die Implementation

Die Transformation von Esterel in SyncChart verläuft in diesen drei Schritten:

1. Benutzung des cec zum Überprüfen der Syntax, des Expandierens des Hauptmoduls und dem Erstellen einer XML basierten Darstellung.

2. Einlesen der XML-Darstellung mit Hilfe von *XPath* in eine *Java* Datenstruktur.

3. Umwandeln der *Java* Datenstruktur in die KIEL Datenstruktur.

6.0.1. Die Verwendung des cec

Zur Erstellung der XML-Darstellung in Form einer `.exp` Datei wird der im Abschnitt 2.4.1 beschriebenen cec. Der cec wird folgendermaßen aufgerufen.

`cec -keep exp -D <Dir> -logfile <Dir><logname> <filepath>`

cec	:	Der Aufruf des cec
–keep exp	:	Löscht nicht den Zwischenschritt der expandierten XML-Darstellung
-D `<DIR>`	:	Das Zielverzeichnis `<DIR>` der erstellten Dateien
–logfile `<Dir><logname>`	:	Erstellt im Zielverzeichnis eine Log Datei mit dem Namen ⟨*logname*⟩
`<filepath>`	:	Der Pfad der zu kompilierenden Esterel Datei

Diese Optionen stellen einen Teil der möglichen Optionen, die in [12] beschrieben sind. Sollte der cec einen Fehler in der Esterel-Datei finden, so wird dieser in der Log-Datei geschrieben. Zudem sind alle Aufrufe des Befehls `run` aus den einzelnen Modulen entfernt worden.

6.0.2. *XPath*

Zum *parsen* der XML-Darstellung wird das *Java* Paket *XPath* verwendet. Dieses Paket stellt eine Beschreibungssprache für XML-Strukturen bereit und ermöglicht so eine Navigation innerhalb eines XML-Dokuments mittels einfacher Zeichenketten. Durch eine solche Zeichenkette, in dem Zusammenhang spricht man vom *XPath*, kann man alle XML-Elemente einer bestimmten Struktur aus dem XML-Dokument herausfiltern. Es gibt in der Sprache *XPath* Schlüsselwörter, wobei nur ein kleiner Teil verwendet wird. Diese Auflistung stellt die hier am meistenverwedeten Schlüsselwörter kurz vor.

`//`	:	Die Wurzel des Dokumentes
`*`	:	alle Nachfolger
`[]`	:	Bedingung
`local-name(.)`	:	Funktion, die die lokalen Namen aller Nachfolger liefert

Eine genauere Auflistung und Erklärung zu *XPath* ist [35] zu entnehmen.

Nachdem nun die Hilfsmittel zum *parsen* einmal vorgestellt wurden wird nun der wichtigste Teil der Implementation, die *Java*-Klassen, erläutert. Im Anschluss daran folgen die Algorithmen zum *parsen* 6.2 und umwandeln 6.3. Im Abschnitt 6.3.2 werden zudem Kompromisse bei der Umwandlung in die *KIEL*-Datenstruktur angegeben.

6.1. Die *Java*-Klassenstruktur für Esterel

Das Ziel beim Aufbau der *Java*-Klassenstruktur für diese Arbeit war es, eine möglichst einfache Wartbarkeit zu erreichen und eine Erweiterung zu ermöglichen. Daher fiel die Entscheidung darauf, den Aufbau der Klassen der Esterel-Grammatik nachzustellen. So besitzt jede Klasse eine Methode zum Einlesen seiner Attribute aus dem XML Dokument und eine Methode zur Umwandlung in die KIEL Datenstruktur. Durch diese Aufteilung ist der Vorgang des Erstellens einer Datenstruktur für ein Esterel Modul und die Umwandlung in eine andere Datenstruktur voneinander getrennt und das jeweilige Vorgehen lässt sich unabhängig verändern. Ein weiterer Vorteil ist, dass dieser Aufbau die Implementation einer Umwandlung in eine andere (als *KIEL*) *Java*-Datenstruktur zulässt.

Es wird allerdings auf den rekursiven Aufbau der Esterel-Befehle verzichtet, sondern die einzelnen Klassen werden mit Indizes, die auf die zugehörigen Befehle in einer `ArrayList` verweisen, ausgestattet. Diese eindimensionale `ArrayList` ist in der Klasse `EsterelModule` 6.1.2, welche ein Esterel Modul repräsentiert, enthalten.

Die *Java*-Klassenstruktur enthält für jeden Esterel-Befehl eine eigene Klasse. Diese Klassen werden von einer abstrakten Superklasse `EsterelStatement` abgeleitet. So enthält jede Klasse, die einen Befehl darstellt, die gleichen Methoden, die die Funktionalität des Einlesens und Umwandelns beinhalten. Nur die Funktionalität muss dem Befehl angepasst werden. So kann man jedes Objekt einer Klasse, die einen Befehl implementiert in der gleiche Art und Weise behandeln, ohne die genaue Art des Befehls zu kennen.

Zunächst werden nun die Superklassen und die Klasse `EsterelModul` beschrieben, da diese für das Verständnis der Implementation von großer Bedeutung sind. Alle Klassen die Esterel-Befehle implementieren, sind im Anhang C in einer UML-Darstellung und im Quelltext dargestellt.

Jede Klasse dieser Datenstruktur enthält einen *Constructor*, der eine Methode, die die entsprechenden Daten aus dem XML-Dokument ausliest, automatisch aufruft.

ht

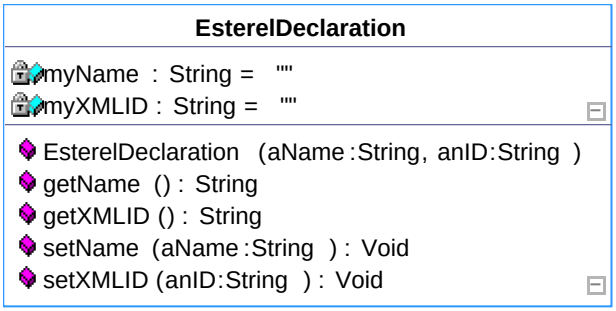


Abbildung 6.1.: UML-Darstellung der Klasse `EsterelDeclaration`

ht

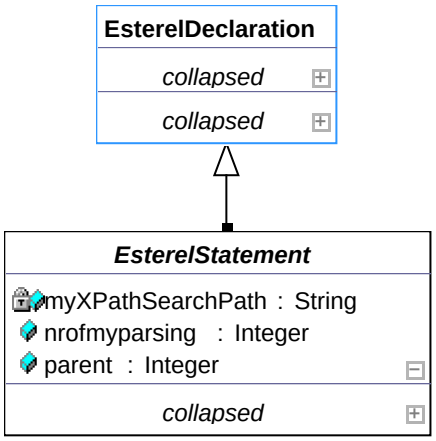


Abbildung 6.2.: UML-Darstellung der Klasse `EsterelStatement`

6.1.1. Die Superklassen

Die Klasse `EsterelDeclaration`

Die noch nicht erwähnte Klasse `EsterelDeclaration` ist die Superklasse, aller für die Umwandlung benötigten Klassen, und beinhaltet nur den Namen und die Identifikationsnummer eines XML-Elementes, sowie Methoden zum Setzen und Lesen der Werte.

Die Klasse `EsterelStatement`

Die Klasse `EsterelStatement`, dargestellt in der Abbildung 6.1.1 ist die Superklasse aller Esterel Klassen, die Befehle implementieren. Sie fügt, den aus der Klasse

vererbten Klassenattributen das Attribut `myXPathSearchString`, welcher auf den entsprechenden Befehl im XML-Dokument verweist, hinzu. Zudem werden die folgenden Methoden implementiert:

- `getEsterelStatementName()` liefert einen eindeutigen Namen, der aus dem Befehlsnamen und der Identifikationsnummmer des XML-Elements besteht, zurück.

- `getXPathSearchString()` liefert den Wert des Klassenattributs zurück.

- `preParseEsterelStatement(EsterelParser)` setzt die Klassenattribute aus dem XML-Dokument.

Desweiteren werden die folgenden abstrakten Methoden zur Verfügung gestellt:

- `parseEsterelStatement(EsterelParser)`,

  Diese Methode liest die Attribute des Befehls aus einem XML-Dokument ein und erhöht die Länge der Liste der Befehle um die Anzahl der im Esterel-Befehl enthaltenden Befehle.

- `setSubStatment(EsterelParser)`

  Diese Methode füllt die zugehörigen Stellen der Liste der Befehle mit den entsprechenden Objekten, wobei der *Constructor* mit dem Aufruf der Methode `parseEsterelStatement` benutzt wird.

- `convertToKiel(EsterelModule, boolean, ArrayList, ArrayList, ArrayList)`

  Diese Methode erstellt entsprechend der Regeln aus Kapitel 3 einen Makrozustand und liefert diesen zurück. Dabei wird per `true` übergeben, dass der Makrozustand ein finaler sei soll. Die ersten beiden `ArrayList`en beinhalten die lokalen Signale und lokalen Variablen, welche bekannt sind. Die letzte `ArrayList` enthält alle übergeordneten `TrapEsterelStatement` Objekte, um einen Zugriff auf die entprechenden Signale zu ermöglichen.

- `toString(EsterelModule)`

  Die `toString` Methode liefert eine Zeichenkette, die das Objekte repräsentiert, zurück.

Die Parameter vom Typ `EsterelParser` und `EsterelModule` werden zum Zugriff auf deren Attribute benötigt.

An dieser Stelle wird zu den Klassen, die die eigentliche Implementation darstellen übergegangen.

ht

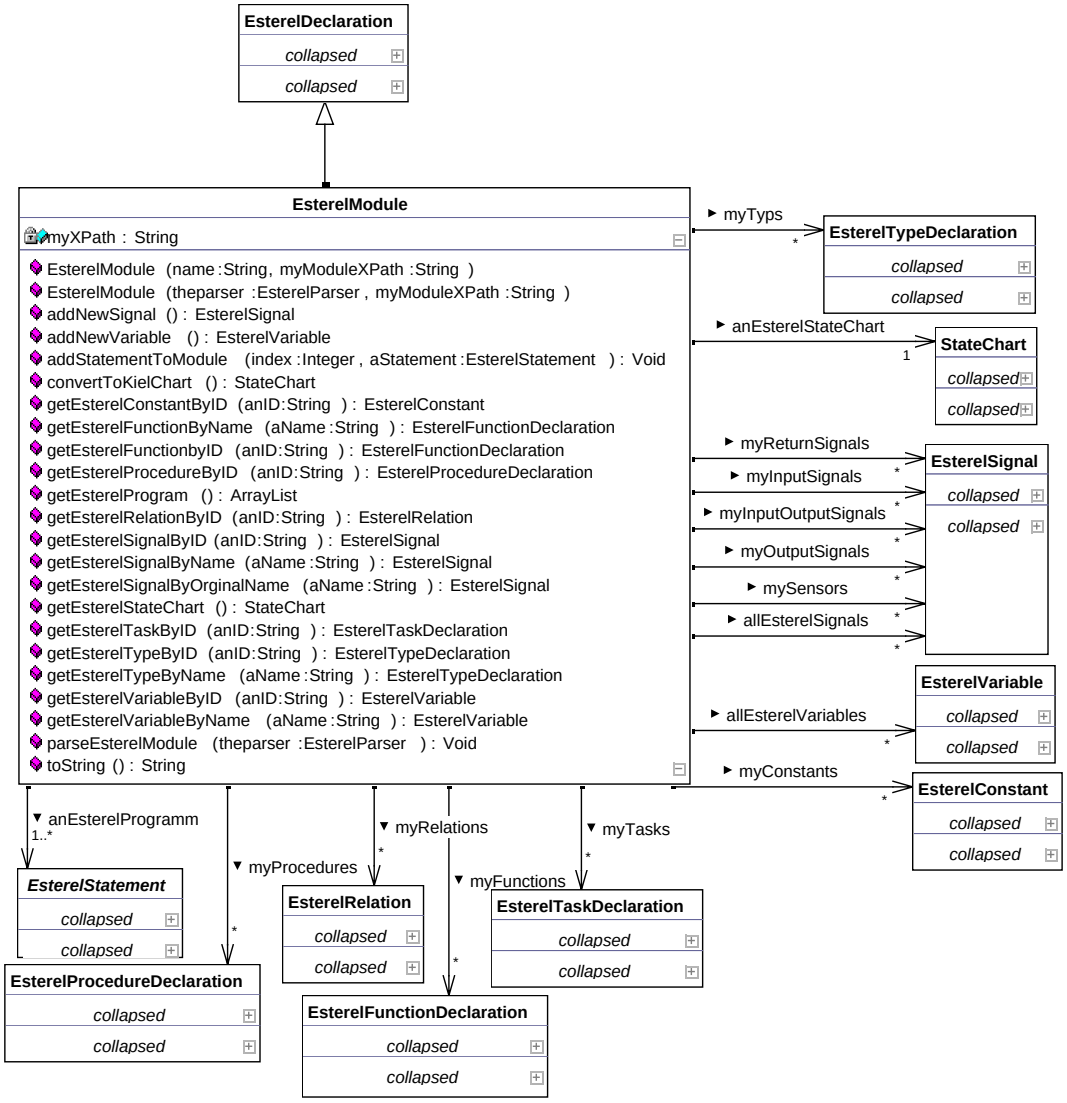


Abbildung 6.3.: UML-Darstellung der Klasse `EsterelModule`

6.1.2. Die Klasse `EsterelModule`

Eine Instanz dieser Klasse enthält, wie in der Abbildung C.2.17 zu sehen, eine `ArrayList` namens `anEsterelProgram`, die alle im Modul vorhandenen Befehle enthält. Dabei steht der Befehl, welcher dem Module durch die Grammatik direkt zugeordnet wird an der Stelle 0. Zudem enthält ein Objekt von `EsterelModule` die gesamten Deklarationen in Form von speziellen Klassen der einzelnen Deklarationsmöglichkeiten wie Signalen, Typen, Funktionen, Prozeduren, Tasks und Relationen. Zur Vereinfachung kennt ein `EsterelModule` alle globalen und lokalen Signale, sowie die lokalen Variablen und globalen Konstanten als Array der entsprechenden Klasse. Dies ermöglicht ein Hinzufügen von neuen zusätzlichen Signalen oder Variablen ohne dass Konflikte in der Namensgebung auftreten können. Mittels der Klassenmethode `convertToKiel` wird die rekursive Umwandlung des Esterel-Moduls in ein SyncChart vorgenommen. Dabei wird das, von der Methode erstellte und zurückgelieferte SyncChart in dem Klassenattribut `anEsterelStateChart` gespeichert.

An dieser Stelle wird das Vorgehen beim Einlesen des gesamten Moduls aus dem XML-Dokument und das Vorgehen bei der Umwandlung vorgestellt.

6.2. *Parsen* eines Esterel Moduls

Der im Rahmen dieser Arbeit entwickelte *Parser* erstellt eine *Java*-Datenstruktur, welche ein Esterel-Modul repräsentiert. Diese Datenstruktur wird im Abschnitt 6.1 beschrieben.

Den Algorithmus des *Parsens* eines Esterel-Moduls, welcher auf der Breitensuche basiert, liefert eine flache Darstellung des Moduls in der entwickelten *Java*- Datenstruktur. Der Algorithmus arbeitet folgendermaßen:

1. Zunächst werden die XPaths für die einzelnen Deklarationen gebildet und durch ein neues Objekt der entsprechenden Klasse eingelesen.

2. Es werden alle lokalen und globalen Signale als Objekte der Klasse `EsterelSignal` erstellt, wobei die Objekte die Signale selbst *parsen*. Dies wird ebenfalls für die Konstanten und alle lokalen Variablen mittels der Klassen `EsterelConstatant` und `EsterelVariable` ausgeführt.

3. Der Zähler `statementCounter` wird auf 0 gesetzt und der Zähler `lastparsedStatement` auf -1. Der *XPath* zum ersten Esterel-Befehl wird ermittelt und in der Variablen `theElementXPathSearchString` gespeichert.

4. An der Position `statementcounter` des `anEsterelProgramm` wird ein Objekt der entsprechenden Klasse, des Esterel-Befehls, auf den `theElementXPathSearchString` verweist, erstellt. Aus den Daten des XML-Dokuments werden an dieser Stelle die eventuell vorhanden Klassenattribute des Objekts selbst erstellt.

5. Der Zähler `lastparsedStatement` wird um 1 erhöht.

6. Solange `statementcounter > lastparsedStatement` ist, wird der folgende Punkt wiederholt, ansonsten ist das Programm vollständig eingelesen.

7. Es wird das Objekt an der Position `lastparsedStatement` betrachtet. Besitzt dieses andere Esterel-Befehle als Attribut, so werden für alle Befehle jeweils nacheinander die folgenden Punkte ausgeführt.

   - Der entsprechende *XPath* wird gebildet und im `theElementXPathSearchString` gespeichert,
   - ss wird wie in Punkt 4 verfahren,
   - in dem entsprechenden Klassenattribut wird der Wert des `statementcounter` vermerkt,
   - der Zähler `statmentcounter` wird um 1 erhöht.

Der Punkt 4 entspricht dem Aufruf der Methode `parseEsterelStatement` und Punkt 7 entspricht der Methode `setSubStatement` aus der Klasse `EsterelStatement`.

Erkennt ein Objekt während des *parsens* Fehler oder unbekannte XML-Strukturen, so wird eine `EsterelParserException` geworfen.

Auf die Beschreibung des *parsens* folgt nun die Beschreibung der Implementation der Transformation.

6.3. Die Transformation in die graphische Darstellung

Hierbei wird wieder die oben genannte Datenstruktur genutzt, unter der jeder Esterel-Befehl seine Umwandlung in ein Makrozustand kennt. Zur Transformation bestanden zwei Möglichkeiten. Zum einen jeden Nicht-Kernelbefehl durch Kernelbefehle zu ersetzen und dann eine graphische Übersetzung der Kernelbefehle zu beschreiben oder zum anderen jeden Befehl einzeln zu übersetzen. An dieser Stelle wurde die zweite Möglichkeit gewählt, da es Umwandlungen in die graphischen Darstellung von Nicht-Kernelbefehle gibt, die sehr leicht zu implementieren sind, wie z. B. `abort` durch den entsprechenden Pfeil, während einige Kernelbefehle, sehr komplex sind.

Der Transformationsalgorithmus basiert auf einer rekursiven Tiefensuche und ist sehr einfach gehalten.

6.3.1. Der Transformationsalgorithmus

Der Transformationsalgorithmus benutzt die `convertToKiel`-Methoden der Klassen für die Esterel-Befehle. Die einzelnen Befehle werden entsprechend der Regeln im Kapitel3 umgewandelt.

1. Zunächst wird aus dem `EsterelModule` ein SyncChart erstellt,

2. dann werden die einzelnen Deklarationen die im `EsterelModule` gespeichert sind, in die *KIEL*-Datenstruktur umgewandelt und in das Statechart eingefügt.

3. Auf das `EsterelStatement` an der Position 0 der Klassenattribute `anEsterel-Programm` wird die Methode `convertToKiel()` angewendet.

4. Besitzt das Objekt von `EsterelStatement` Verweise auf weitere `Esterel-Statement`s, so wird auf diese die Methode `convertToKiel()` angewendet. Für jeden Verweis wird dieser Punkt ausgeführt.

5. Besitzt das `EsterelStatement` keinen Verweis, oder es sind alle Verweise abgearbeitet so wird der erstellte Makrozustand zurückgeliefert und entsprechend in das Statechart eingefügt.

6.3.2. Kompromisse bei der Transformation in *KIEL*

Bei der Umwandlung in die *KIEL*-Datenstruktur mussten aus verschieden Gründen Kompromisse bei der Transformation gemacht werden.

- Signale, die als *InputOutput* deklariert sind, werden in *KIEL* nur als Input Signal erstellt.
- Es können nur Konstaten und Variablen der Typen *Float* und *Double* genutzt werden, Ausdrücke werden momentan noch nicht unterstützt.
- Die Aufrufe von `call, exec` werden angezeigt, können allerdings intern nicht verarbeitet werden, da die Funktionalität in einer *Host*-Sprache implementiert wird.
- Funktionen werden als Variable implementiert und können so innerhalb von *KIEL* verwendet werden. Ihre Funktionalität müsste eingentlich in einer *Host* Sprache implementiert werden.
- Konstante werde auch als lokale Variable im Wurzelzustand implementiert.
- Typen, die nicht `integer, boolean, double, float` oder `string` heißen können nicht benutzt werden, da die Funktionalität in einer *Host*-Sprache implementiert wird. Die Verwendung wird als Zeichenkette angezeigt. Anwendungen von Komponeten dieser Typen werden als Zeichenkette implementiert.
- Relationen werden im Simulator nicht berücksichtigt.
- Da von *KIEL* nur ein Statechart zur Zeit bearbeitet werden kann, wird aus Esterel-Dateien die mehrere Module, in denen kein `run`-Aufruf steht, nur das textlich erste Module transferiert. Es werden aber alle Module eingelesen.

6.4. Optimierung

Die Regeln zur Optimierung eines SyncChart und den Algorithmus werden in Kapitel 4 beschrieben. Bei der Vereinfachung eines Statecharts gibt es die Möglichkeit zwischen einer schrittweisen und einer kompletten Vereinfachung.

Die einzelnen Regel wurden im Paket `kiel.optimizer.esterelstudio` jeweils als Klasse implementiert. Die Abbildung C.4 stellt die Klassenstrucktur dar. Die Superklasse der Regeln `OptimizerRule` definiert die Methoden `isOptimizationPossible` und `optimize` abstrakt, so dass diese beiden Methoden sind in den einzelnen Klassen entsprechend zu definieren sind. Die Methode `isOptimizationPossible` überprüft, ob die Regel angewendet werden kann, während die Methode `optimize` die Regel auf einen Makrozustand anwendet. Sollte eine Regel erfolgreich angewendet werden, so wird ein `true` zurückgeliefert. Anderenfalls wird ein `false` zurückgeliefert oder es wird eine `OptimizerException` in dem Fall einer Ausnahme geworfen.

Die im Kapitel 4 beschriebenen Regeln erläuterten nur jeweils die Vereinfachung eines Makrozustandes. Im Folgenden wird nun die Vereinfachung eines gesamten SyncChart unter Verwendung der Regeln aufgezeigt.

Da ein Makrozustand nur dann endgültig vereinfacht werden kann, wenn alle internen Zustände bereits in einer vereinfachten Form vorliegen, bietet es sich an an der untersten Ebene mit der Vereinfachung zu beginnen und die Makrozustände ebenenweise abzuarbeiten. So muß jeder Makrozustand nur einmal betrachtet werden.

Definition 14 (Ebene im SyncChart) Die Ebene, in der ein Zustand liegt, entspricht der Anzahl der ihn umgebenden Makrozustände bis zum in der Hierarchie obersten Zustand.

Der Algorithmus, der ein gesamtes SyncChart vereinfacht, setzt voraus, dass es ein Feld `myChart` mit Verweisen zu den Makrozustände des SyncCharts gibt, welches aufsteigend nach der Ebene der Makrozustände sortiert ist. Zudem bedarf es des Parameters `all` der angibt, ob schrittwise oder komplett vereinfacht werden soll. Dazu wird ein Übergabeparameter `all` angezeigt. Ist `all = true`, so wird das SyncChart komplett vereinfacht, ansonsten wird nach der ersten erfolgreichen Ausführung einer Regel die Vereinfachung abgebrochen. Die Regel sind in einem Feld `f` entsprechend ihrer Ausführungs-Reihenfolge im Feld sortiert.

1. Setze `i` auf Index des letzten Eintrages in `myChart`.
2. Setze die boolsche Variable `aNoChange` mit den Wert von `all`.
3. Solange `i>-1` fahre mit den Punkten a),b) und c) fort.
   a) Setze den Zähler `ruleCounter` auf 0.
   b) Solange `ruleCounter` kleiner als die Länge des Feldes `f` und `aNoChange = true` ist, wende die Regel, auf welche `ruleCounter` verweist, auf den Zustand mit dem Index `i` aus `myChart` an. `aNoChange` wird auf den Rückgabewert der Regel `ruleCounter` Oder-verknüpft mit `all` gesetzt und `ruleCounter` um eins erhöht.
   c) Setze `i` auf `i-1`.
4. Liefer das optimierte SyncChart zurück.

Dieser Algorithmus ist in der Klasse `EsterelStudioChartOptimizer` in der Methode `startOptimize` implementiert, welche durch die Superklasse `Optimizer` abstrakt definiert ist.

Der *Java*-Programmcode zu den beschriebenen Implemtierungen ist im Anhang C dargestellt.

7. Zusammenfassung und Ausblick

Zum Abschluss dieser Arbeit werden die Ergebnisse wiederholt und ein Ausblick gegeben.

7.1. Zusammenfassung

Die in Kapitel 3 erstellten Transformationsregeln für die einzelnen Esterel-Befehle, ermöglichen die theoretische Transformation von Esterel nach SyncChart. Dadurch, dass die Transformationsregeln jeden Esterel-Befehl, mit Ausnahme von `run` und `exec`, in jeweils einen Makrozustand umwandeln, entstehen in der Regel unnötige Makrozustände. Diese Makrozustände sollten daher aus dem SyncChart entfernt werden, um eine verständlichere Darstellung zu erhalten. Aus diesem Grund wurden zusätzlich im Kapitel 4 Regeln zum Optimieren eines SyncCharts angegeben. Diese Regeln dienen nur der strukturellen Optimierung eines SyncCharts, das Verhalten des SyncCharts wird dabei nicht verändert. Durch die in Kapitel 6 beschriebene Implementation der Transformations-und der Optimierungsregeln in das Projekt *KIEL*, ist die praktische Transformation von Esterel-Programmen nach SyncChart. Ein Beispiel ist im Kapitel 5 aufgeführt.

Die Anwendungsmöglichkeiten für die Ergebnisse diese Arbeit sind vielseitig und können sowohl als Dokumentation für Esterel-Programme dienen oder zur grafischen Simulation von Esterel-Programmen genutzt werden. Das wichtigste Anwendungsgebiet ist aber die schnelle Erstellung eines reaktiven Systems durch die textliche Sprache in einem Texteditor und die weitere Verwendung des Systems in einem grafischen Modellierungswerkzeug.

7.2. Ausblick

Diese Arbeit dient als Grundlage zur Transformation von Esterel nach SyncChart. Eine Erweiterung für Esterel-Versionen nach „Esterel v5 “sollte möglich sein. Interessanter sind jedoch Erweiterungen in *KIEL*, um insbesondere die im Abschnitt 6.3.2 aufgeführten Kompromisse zu Gunsten der Transformation zu lösen. Eine weitere Verbesserung wäre die Möglichkeit aus der *KIEL*-Datenstruktur Esterel-Programmcode zu erzeugen, da so die Erstellung und Bearbeitung von SyncCharts vollständig über eine textlich Programmiersprache möglich wäre. Desweiteren wäre es möglich, analog zu den hier entwickelten Transformationsregeln Transformationen in andere Statechart-Dialekte zu erstellen.

Die Optimierung von SyncCharts kann ebenfalls für weitere Dialekte in *KIEL* integriert werden. Sollte die Optimierung auf die Semantik erweitert werden, so würde die Optimierung komplettiert werden.

A. Literaturverzeichnis

[1] Charles André. SyncCharts: A Visual Representation of Reactive Behaviors. Technical Report RR 95–52, rev. RR (96–56), I3S, Sophia-Antipolis, France, Rev. April 1996. URL `http://www.i3s.unice.fr/~andre/CAPublis/SYNCCHARTS/SyncCharts.pdf`.

[2] Charles André. Computing SyncCharts Reactions. In *Electronic Notes in Theoretical Computer Science*, volume 88. Elsevier, July 2003. URL `http://www.inrialpes.fr/bip/people/girault/Slap03/Final/andre.pdf`.

[3] Charles André. Semantics of S.S.M (Safe State Machine). Technical report, I3S, Sophia-Antipolis, France, 2003. URL `http://www.esterel-technologies.com/v3/?id=50399\&dwnID=48`.

[4] Gerard Berry. *The Constructive Semantics of Pure Esterel*. Draft Book, 1999. URL `ftp://ftp-sop.inria.fr/esterel/pub/papers/constructiveness3.ps`.

[5] Gerard Berry. *The Esterel v5 Language Primer*, 1999. URL `ftp://ftp-sop.inria.fr/meije/esterel/papers/primer.ps`.

[6] Gerard Berry. The foundations of Esterel. *Proof, Language and Interaction: Essays in Honour of Robin Milner*, 2000. Editors: G. Plotkin, C. Stirling and M. Tofte.

[7] Gerard Berry and Georges Gonthier. The Esterel Synchronous Programming Language: Design, Semantics, Implementation. *Science of Computer Programming*, 19(2):87–152, 1992. URL `http://citeseer.nj.nec.com/berry92esterel.html`.

[8] CEC. CEC: The Columbia Esterel Compiler. URL `http://www1.cs.columbia.edu/~sedwards/cec/`.

[9] Noam Chomsky. Three models for the description of language. In *IRE Transactions on Information Theory 2*, pages 113–124, 1956.

[10] Noam Chomsky. On certain formal properties of grammars. In *Information and Control 1*, pages 91–112, 1959.

[11] Christian-Albrechts-Universität zu Kiel. Christian-Albrechts-Universität zu Kiel. URL `http://www.uni-kiel`. Christian-Albrechts-Universität zu Kiel 24098 Kiel, Germany.

[12] Columbia University. cec manpage. URL `http://www1.cs.columbia.edu/~sedwards/cec/cec.pdf`.

[13] Duden. *Duden - Das groše Wörterbuch der deutschen Sprache in zehn Bänden*. Dudenverlag, 2005. ISBN ISBN 3-411-70360-1.

[14] Stephen A. Edwards. CEC: The Columbia Esterel Compiler. `http://www1.cs.columbia.edu/~sedwards/cec/`.

[15] Esterel Technologies. Free Sofware Esterel Compiler. URL `http://www.esterel-technologies.com/technology/free-software/esterel-compiler.html`. Esterel Compiler.

[16] *Esterel Studio User Guide and Reference Manual*. Esterel Technologies, 5.0 edition, May 2003.

[17] S. Moisan G. Berry and J-P. Rigault. Esterel: Towards a synchronous and semantically sound high-level language for real-time applications. In *IEEE Real-Time Systems Symposium*, volume IEEE Catalog 83CH1941-4, pages 30–40, 1983.

[18] Grégoire Hamon and John Rushby. An operational semantics for Stateflow. In M. Wermelinger and T. Margaria-Steffen, editors, *Fundamental Approaches to Software Engineering (FASE)*, volume 2984 of *Lecture Notes in Computer Science*, pages 229–243, Barcelona, Spain, April 2004. Springer.

[19] David Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, June 1987.

[20] Jeffrey D. Ullman John E. Hopcroft. Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie], 1990. ISBN ISBN 3-929821-03-6.

[21] Tobias Kloss. Flexibles und Automatisiertes Layout von Statecharts. Studienarbeit, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik, July 2003.

[22] Tobias Kloss. Automatisches Layout von Statecharts unter Verwendung von GraphViz. Diplomarbeit, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik, May 2005.

[23] Florian Lüpke. Implementierung eines Statechart-Editors mit layoutbasierten Bearbeitungshilfen. Diplomarbeit, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik, June 2005.

[24] Markus Kuhn. ISO/IEC 14977:1996(E). URL `http://www.cl.cam.ac.uk/~mgk25/iso-14977.pdf`.

[25] Mathworks Inc. *Stateflow and Stateflow Coder for use with Simulink — User's Guide*. Mathworks Inc., 6 edition, 2004.

[26] André Ohlhoff. Simulating the Behavior of Syncharts. Studienarbeit, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik, November 2004.

[27] Dieter Maurer Reinhard Wilhelm. *Übersetzerbau - Theorie, Konstruktion, Generierung*. Springer, 1997. ISBN ISBN 3-540-61692-6.

[28] Sanjit A. Seshia, R. K. Shyamasundar, A. K. Bhattacharjee, and S. D. Dhodapkar. A translation of statecharts to esterel. In J. Wing, J. Woodcock, and J. Davies, editors, *FM'99 volume 2— World Congres on Formal Methods*, volume 1709 of *LNCS*, pages 983–1007. Springer-Verlag, 99.

[29] Stephen A. Edwards. Stephen A. Edwards Homepage. URL `http://www.cs.columbia.edu/sedwards/`. Esterel Compiler.

[30] Sun Developer Network. Java Homepage. URL `http://java.sun.com/`.

[31] The KIEL Project. Project API Documentation, 2004. URL `http://www.informatik.uni-kiel.de/~rt-kiel/kiel/kiel/doc/`. Kiel Integrated Environment for Layout.

[32] The KIEL Project. Project Homepage, 2004. URL `http://www.informatik.uni-kiel.de/~rt-kiel/`. Kiel Integrated Environment for Layout.

[33] The Object Management Group. UML Homepage. URL `http://www.uml.org/`.

[34] Mirko Wischer. Ein FileInterface für das KIEL Projekt. Praktikumsbericht, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik, 2005.

[35] World Wide Web Consortium (W3C) . XPath Homepage. URL `http://www.w3.org/TR/xpath`. Extensible Markup Language (XML).

B. Bedienungsanleitung

B.1. Transformation von Esterel nach SyncChart

Um eine Transformation von Esterel in ein SyncChart vornehmen zu können, müssen die folgenden Vorraussetzungen erfüllt sein:

- *Java* muss zur Verfügung stehen,
- *KIEL* muss auf dem System vorhanden sein,
- der cec muss ist in *KIEL* vorhanden.

Die Umwandlung wird automatisch beim Einlesen einer beliebigen `.strl`-Datei gestartet. Dabei macht es keinen Unterschied ob man die GUI von *KIEL* benutzt oder das Programm per Kommandozeile startet. Das Verhalten der Umwandlung kann über zwei Dateien beeinflußen, welche im Verzeichnis `/.kiel` vorhanden sind. Sollten man Veränderungen an diesen Dateien vorgenommen werden, kann es passieren das die korrekte Ausführung des Programmes verhindert wird.

B.1.1. Die Datei `esterel.properties`

In dieser Datei können verschiedene Einstellung verändert werden, die nicht direkt mit der Umwandlung im Zusammenhang stehen.

AutomaticOptimizationOfEsterelStateCharts Hier hat man die Wahl zwischen `true` und `false`. `true` gibt an, dass man SyncChart ab, einer später zu definierenden Größe, sofort vereinfacht werden, `false` verbietet diese automatische Optimierung.

NumberOfStatesToStartWithOptimization Die Zustandsanzahl ab der gegebenenfalls ein SyncChart automatisch optimiert wird.

cecLogfileName Der Name der Datei, in der die Meldungen des letzten Aufrufs des cec stehen.

cecWinPath ist der Pfad zum cec unter Windows,

cecLinuxPath ist Pfad zum cec unter Linux.

cygWinDir ist der Pfad zu *cygwin*. Dieser muss nur bei Windows gesetzt sein.

cygWinBinPath ist der Pfad zum `bin`-Verzeichnis unter *cygwin*.

DebugMode Bei `true` werden verschiedene Meldungen auf dem Bildschirm ausgegeben. Bei `false` werden keine Meldungen erzeugt.

B.1.2. Die Datei `esterel2estudio.properties`

In dieser Datei kann man verschiedene Einstellung ändern, die sich direkt bei der Umwandlung ins SyncChart auswirken.

PreSting Diese Zeichenkette wird am Anfang von jedem, bei der Umwandlung hinzugefügten, Signalnamen angefügt.

PostString Diese Zeichenkette wird am Ende jedes, bei der Umwandlung hinzugefügten, Signalnamens angefügt.

Functions Bei `true` werden Funktionen so umgewandelt, dass eine Simulation in *KIEL* möglich ist. Funktionen werden dann als Variablen gleichen Namens implementiert, bei `false` wird die Funktion im Orginalzustand belassen. Eine Simulation in *KIEL* ist unwahrscheinlich.

Procedures Bei `true` werden Prozeduraufrufe nicht generiert, bei `false` werden Prozeduraufrufe umgewandelt. Eine Simulation in *KIEL* ist unwahrscheinlich.

Tasks Bei `true` werden Taskaufrufe nicht gerneriert, bei `false` werden Taskaufrufe umgewandelt. Eine Simulation in *KIEL* ist ebenfalls unwahrscheinlich.

ArgumentSeperator Ein Zeichen, welches die Übergabeparameter in einem Funktions, Prozedur-oder Taskaufruf voneinander trennt.

B.2. Der Optimizer

Der Optimizer geht davon aus, dass ihm ein korrektes SyncChart übergeben wird. SyncCharts können wahlweise komplett oder regelweise vereinfacht werden. Diesen Vorgang kann in der GUI über den Menüpunkt Optimizer, durch die dort angegebene Tastenkombination auslöst werden. Mit Hilfe des Kommandozeilen Befehls `kielcmd` kann die komplette Optimierung mit der Option `-Opt` veranlassen werden.

Es können verschieden Parameter bei der Optimierung verändert werden. Diese Einstellungen kann man in der Datei `optimizer.properties` im Verzeichnis `/.kiel` verändern.

DebugMode Bei `true` werden verschiedene Meldungen auf dem Bildschirm ausgegeben. Bei `false` werden keine Meldungen erzeugt.

OptimizeESTEliminateNeedlessConditional aktiviert die Regel zum Entfernen unnötiger *Conditional*-Pseudozustände über `true`, wird `false` gesetzt, so wird die Regel deaktiviert.

OptimizeESTEliminateNeedlessNormalTransitions aktiviert die Regel zum Entfernen unnötiger *normal termination*über `true`, wird `false` gesetzt, so wird die Regel deaktiviert.

OptimizeESTEliminateNeedlessSimpleStates aktiviert die Regel zum Entfernen von unnötiger, einfacher Zustände über `true`, wird `false` gesetzt, so wird die Regel deaktiviert.

OptimizeESTEliminateNotAbortednWithoutLocalsOrStates aktiviert die Regel zum Auflösen von Zuständen, die lokale Signale oder Variablen definieren, über `true`, wird `false` gesetzt, so wird die Regel deaktiviert.

OptimizeESTJoinFinalStates aktiviert die Regel zum Enfernen von unnötiger finaler Zuständen über `true`, wird `false` gesetzt, so wird die Regel deaktiviert.

OptimizeESTUpdateFinalStates aktiviert die Regel zum Umwandeln finaler in nichtfinale Zustände über `true`, wird `false` gesetzt, so wird die Regel deaktiviert.

OptimizeESTOrStatesWithTwoSubstates aktiviert die Regel zum Entfernen von Makrozuständen mit nur zwei untergeordneten Zuständen über `true`, wird `false` gesetzt, so wird die Regel deaktiviert.

C. Java Code für die Transformation von Esterel in SyncCharts

Der im Folgenden aufgelistete *Java*-Programmcode wurde um die Fehler bereinigt, welcher der *Java*-Compiler, der *Java*-Dokumentation-Generator und die Werkzeuge *JLint* und *Check Style* mit den *Sun-Coding-Conventions* liefern. Der vorliegende *Java*-Programmcode ist mit der *Java*-Version 1.4 übersetzbar. Bei der Darstellung der einzelnen *Java*-Klasse mittels UML, wird nur der Ausschnitt des Befehls dargestellt und auf Relationen der anderen dargestellten Klassen verzichtet, um die Übersichtlichkeit zu wahren.

C.1. Paket `kiel.fileInterface.esterel`

Das Paket `kiel.fileInterface.esterel` enthält die Klassen zur Implementation des *FileInterface* für Esterel-Dateien benötigt werden.

C.1.1. EsterelFileFilter

Auflistung C.1: Die Klasse `EsterelFileFilter`

```
package kiel.fileInterface.esterel;
import java.io.File;
import javax.swing.filechooser.FileFilter;
/**
 * <p> A Esterel Plugin for the generic fileinterface</p>
 * <p><b>Details on this special class:</b>his implements the Interface all
 * Fileformat Parser/Generator must implement.
 * </p>
 * <p>Copyright: Copyright (c) 2004</p>
 * <p>Company: Uni Kiel</p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
 * @version $Revision: 1.28 $ last modified $Date: 2006/02/06 18:35:28 $
*/
public class EsterelFileFilter extends FileFilter {
    /** an intance counter.*/
    private static EsterelFileFilter instance = null;
    /**
     *Saves that only one instace is active.
     * @return an instace of
     * OriginalEsterelFileFilter
     */
    public static EsterelFileFilter getInstance() {
        if (instance == null) {
            instance = new EsterelFileFilter();
        }
        return instance;
    }
    /**
     *Checks if the path is correct.
     *<br>Sideeffects:
     *none
     * @param pathname a pathname
     * @return <code>true</code> if correct else <code>false</code>
     */
    public final boolean accept(final File pathname) {
        if (pathname.isDirectory()) {
            return true;
        }
        return (
            pathname.canRead()
                && pathname.isFile()
                && pathname.getPath().toLowerCase().endsWith(".strl"));

    }
    /**
     * @return "Esterel Files (as ast with the cec)"
     */
    public final String getDescription() {
        return "Esterel Files .strl";
    }
}
```

C.1.2. Esterel

Auflistung C.2: Die Klasse `Esterel`

```
package kiel.fileInterface.esterel;

//
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.ArrayList;
import javax.swing.filechooser.FileFilter;
import kiel.dataStructure.CompositeState;
import kiel.dataStructure.StateChart;
import kiel.fileInterface.FileInterface;
import kiel.fileInterface.FileInterfaceException;
import kiel.fileInterface.esterel.esterel2estudio.Esterel2EstudioException;
import kiel.fileInterface.esterel.esterel2estudio.Esterel2EstudioProperties;
import kiel.fileInterface.esterel.esterel2estudio.EsterelModule;
import kiel.graphicalInformations.View;
import kiel.optimizer.OptimizerChooser;
import kiel.optimizer.OptimizerException;

import org.jdom.Document;
import org.jdom.JDOMException;
import org.jdom.input.SAXBuilder;

/**
 * <p>
 *  A Esterel Plugin for the generic fileinterface.
 * </p>
 * <p>
 * <b>Details on this special class: </b>his implements the Interface all
 * Fileformat Parser/Generator must implement.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2004
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.65 $ last modified $Date: 2006/02/07 17:57:36 $
 */
public class Esterel
       extends FileInterface {
   /**
    * copies a file.
    *
    * @param src
    *                source
    * @param dest
    *                destination
    * @param bufSize
    *                buffersize
    * @param force
    *                true
    * @throws IOException
    *                an exception
    */
   public static void copyFile(
          final File src,
          final File dest,
          final int bufSize,
          final boolean force)
          throws IOException {
      if (dest.exists()) {
          if (force) {
              dest.delete();
          } else {
              throw new IOException(
                     "Cannot overwrite existing file: "
                            + dest.getName());
          }
      }
      byte[] buffer = new byte[bufSize];
      int read = 0;
      InputStream in = null;
      OutputStream out = null;
      try {
          in = new FileInputStream(
                 src);
          out = new FileOutputStream(
                 dest);
          while (true) {
              read = in.read(buffer);
              if (read == -1) {
                  //-1 means EOF
                  break;
              }
              out.write(buffer,
                  0,
                  read);
          }
      } finally {
          //close streams
          if (in != null) {
              try {
                  in.close();
              } finally {
                  if (out != null) {
                      out.close();
```

```
                }
            }
        }
    }
}

/**
 * aLayouter which layouts the statechart.
 */
//private Layouter aLayouter = null;
/**
 * Returns true.
 *
 * @return <code>true</code>
 */
public final boolean canRead() {
    // TODO Auto-generated method stub
    return true;
}

/**
 * Returns false.
 *
 * @return <code>false</code>
 */
public final boolean canWrite() {
    return false;
}

/**
 * Retruns Esterel Plugin.
 *
 * @return "Esterel Plugin"
 */
public final String getName() {
    // TODO Auto-generated method stub
    return "Esterel Plugin";
}

/**
 * Returns null.
 *
 * @return <code>null</code>
 * @throws FileInterfaceException
 *                 not used
 */
public final View getReadView()
        throws FileInterfaceException {
    View av = null;
    return av;
}

/**
 * Returns an instance of <code>OriginalEsterelFileFilter</code>.
 *
 * @return an instace of <code>OriginalEsterelFileFilter</code>
 */
public final FileFilter getStateChartFileFilter() {
    return EsterelFileFilter.getInstance();
}

//public final StateChart runUnderWindows(){}
/**
 * <p>
 * Parses a file <code> f </code> which is an .exp file created by the
 * cec and returns a KIEL statechart representation of the file.
 * </p>
 * <br>
 * Sideeffects: none
 *
 * @param f
 *                a file
 * @return a Kiel statechart
 * @throws FileInterfaceException
 *                 if something goes wrong.
 * @see kiel.dataStructure.StateChart
 */
public final StateChart readStateChartDocument(
        final File f)
        throws FileInterfaceException {
    EsterelProperties.load();
    Esterel2EstudioProperties.load();
    final int aBUFFER = 1024;
    String theExceptions = "";
    StateChart aStateChartOfAnEsterelfile = null;
    EsterelModule[] anEsterelModule = null;
    File theexpfile = null;
    File thecFile = null;
    String expName = null;
    String cName = null;
    String expDir = null;
    final String separator = File.separator;
    String os = System.getProperty("os.name");
    os = os.toLowerCase();
    if (os.compareTo("linux") != 0) {
        os = "windows";
    }
    String pathToCec = EsterelProperties.getCecPath();
    if ((f.canRead()
            && f.isFile() && f.getPath().toLowerCase().endsWith(".exp"))) {
        theexpfile = f;
    } else if ((f.canRead()
            && f.isFile() && f.getPath().toLowerCase().endsWith(".strl"))) {
        expDir = System.getProperty("user.home")
                + separator + ".kiel" + separator;
        expName = f.getName().trim().subSequence(0,
            f.getName().trim().lastIndexOf("."))
                + ".exp";
        cName = f.getName().trim().subSequence(0,
            f.getName().trim().lastIndexOf("."))
                + ".c";
        String filepath = f.getAbsolutePath();
```

```
if (os.compareTo("windows") == 0) {
    // Make it work under cygwin
    File dummy = new File(
           "dummy.strl");
    String runCecCommand = "export PATH="
           + EsterelProperties.getBinPathCygWin() + ":$PATH\n"
           + pathToCec.replace('\\',
              '/') + "cec --keep exp " + "--logfile "
           + EsterelProperties.getNameOfLogfile()
           + " dummy.strl";
    File callSkript = new File(
           "runcec.sh");
    File adummyc = new File(
           "dummy.c");
    File aExpdummy = new File(
           "dummy.exp");
    File aExpKiel = new File(
           expDir
                  + expName);
    File aLog = new File(
           EsterelProperties.getNameOfLogfile());
    File aKielLog = new File(
           expDir
                  + EsterelProperties.getNameOfLogfile());
    File bat = new File(
           "startcec.bat");
    try {
        copyFile(f,
           dummy,
           aBUFFER,
           true);
        if (callSkript.exists()) {
           callSkript.delete();
        }
        callSkript.createNewFile();
        FileWriter fw = new FileWriter(
              callSkript);
        fw.write(runCecCommand);
        fw.flush();
        fw.close();
        String execRunCec = "cd "
              + callSkript.getAbsolutePath().substring(0,
                 callSkript.getAbsolutePath()
                        .lastIndexOf(File.separator))
              + System.getProperty("line.separator")
              + EsterelProperties.getCygwinPath()
              + File.separator + "bash "
              + callSkript.getName();
        if (bat.exists()) {
           bat.delete();
        }
        bat.createNewFile();
        FileWriter batfw = new FileWriter(
              bat);
        batfw.write(execRunCec);
        batfw.flush();
        batfw.close();
        Process q = Runtime.getRuntime().exec("startcec.bat");
        q.waitFor();
        callSkript.delete();
        copyFile(aExpdummy,
           aExpKiel,
           aBUFFER,
           true);
        copyFile(aLog,
           aKielLog,
           aBUFFER,
           true);
    } catch (IOException anIOException) {
        theExceptions += anIOException.getMessage()
              + System.getProperty("line.separator");
    } catch (InterruptedException e) {
        theExceptions += e.getMessage()
              + System.getProperty("line.separator");
    } finally {
        if (adummyc.exists()) {
           adummyc.delete();
        }
        if (aExpdummy.exists()) {
           aExpdummy.delete();
        }
        if (aLog.exists()) {
           aLog.delete();
        }
        if (dummy.exists()) {
           dummy.delete();
        }
        if (callSkript.exists()) {
           callSkript.delete();
        }
        if (bat.exists()) {
           bat.delete();
        }
    }
} else {
    try {
        String run = pathToCec
              + "cec --keep exp " + "-D " + expDir + " "
              + "--logfile " + expDir
              + EsterelProperties.getNameOfLogfile() + " "
              + filepath;
        Process p = Runtime.getRuntime().exec(run);
        p.waitFor();
    } catch (IOException anIOException) {
        theExceptions += anIOException.getMessage()
              + System.getProperty("line.separator");
    } catch (InterruptedException e) {
        theExceptions += e.getMessage()
              + System.getProperty("line.separator");
    }
}
theexpfile = new File(
```

```
                expDir
                        + expName);
330         } //if strl
            try {
                if (theexpfile != null
                        && theexpfile.canRead()) {
                    SAXBuilder builder = new SAXBuilder();
                    builder.setExpandEntities(false);
                    Document doc = builder.build(theexpfile.getAbsoluteFile());
                    try {
                        EsterelParser aparser = null;
                        try {
340                         aparser = new EsterelParser(
                                    doc);
                            anEsterelModule = aparser.gettheEsterelModules();
                        } catch (EsterelParserException epe) {
                            //epe.printStackTrace();
                            theExceptions += epe.getMessage()
                                    + System.getProperty("line.separator");
                            aparser = null;
                        }
                        if (EsterelProperties.getDebugMode()
350                             &&  anEsterelModule != null) {
                            System.out.println(anEsterelModule[0].toString());
                        }
                        if (aparser != null) {
                            try {
                                    if (anEsterelModule.length > 0) {
                                    aStateChartOfAnEsterelfile =
                                        anEsterelModule[0].convertToKielChart();
                                    }
                            } catch (Esterel2EstudioException e) {
360                             e.printStackTrace();
                                theExceptions += e.getMessage()
                                        + System.getProperty("line.separator");
                            }
                        } //if
                        if (f.canRead()
                                && f.isFile()
                                && f.getPath().toLowerCase().endsWith(".strl")) {
                            theexpfile.delete();
                            thecFile = new File(
370                                 expDir
                                            + cName);
                            if (thecFile.exists()) {
                                thecFile.delete();
                            }
                        }
                    } catch (java.lang.StackOverflowError e) {
                        theExceptions += e.getMessage()
                                + System.getProperty("line.separator");
                    }
                } else {
380                 theExceptions += "The .exp file was not created."
                            + System.getProperty("line.separator")
                            + "Please check .kiel" + File.separator
                            + "cec.log for more informations."
                            + System.getProperty("line.separator");
                }
            /*} catch (NullPointerException e) {
                e.printStackTrace();
                theExceptions += e.getMessage()
390                     + System.getProperty("line.separator");*/
            } catch (SecurityException e) {
                theExceptions += e.getMessage()
                        + System.getProperty("line.separator");
            } catch (JDOMException e) {
                theExceptions += e.getMessage()
                        + System.getProperty("line.separator");
            } catch (IOException e) {
                theExceptions += e.getMessage()
                        + System.getProperty("line.separator");
400         } finally {
                File exp = new File(
                        expDir
                                + expName);
                if (exp.exists()) {
                    exp.delete();
                }
                File c = new File(
                        expDir
                                + cName);
410             if (c.exists()) {
                    c.delete();
                }
            }
            if (theExceptions != "") {
                //System.out.println(theExceptions);
                throw new FileInterfaceException(
                        theExceptions);
            }
            if (aStateChartOfAnEsterelfile != null
420                 && EsterelProperties.getAutomaticOptimization()
                    && checkNumberOfStates(aStateChartOfAnEsterelfile.getRootNode())
                    > EsterelProperties.getNrOfStatesForAutomaticOptimization()) {
                try {
                return OptimizerChooser.startOptimize(aStateChartOfAnEsterelfile,
                    true);
                } catch (OptimizerException ex) {
                 throw new FileInterfaceException(ex.getMessage());
                }
            }
430         return aStateChartOfAnEsterelfile;
        }

        /**
         * Not implemented yet.
         *
         * @see kiel.fileInterface.FileInterface#writeStateChartDocument
         *          (kiel.dataStructure.StateChart,
         *          kiel.graphicalInformations.View, java.io.File)
         */
```

```
440     public final File writeStateChartDocument(
                final StateChart s,
                final View v,
                final File f) {
            // TODO Auto-generated method stub
            return null;
        }

        /**
         * counts recusive the number of states.
         *
450      * @param aCState    a CompositeState
         * @return the number of substates.
         */
        private int checkNumberOfStates(
                final CompositeState aCState) {
            ArrayList subnodes = aCState.getSubnodes();
            int result = subnodes.size();
            for (int i = 0; i < subnodes.size(); i++) {
                if (subnodes.get(i) instanceof CompositeState) {
                    result += checkNumberOfStates((CompositeState) subnodes.get(i));
460                 }
            }
            return result;
        }
    }
```

C.1.3. EsterelParserException

Auflistung C.3: Die Klasse `EsterelParserException`

```
package kiel.fileInterface.esterel;
//import kiel.fileInterface.FileInterfaceException;
import javax.swing.JOptionPane;
/**
  * <p> The Exceptionclass for the EsterelParser. </p>
 * <p>Copyright: Copyright (c) 2005</p>
 * <p>Company: Uni Kiel</p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
 */
public class EsterelParserException extends /*FileInterface*/Exception {
    /**
     * the parser state in the moment of the exception.
     */
    private EsterelParser myEsterelParser;
    /**
     * the standart constructor.
     */
    public EsterelParserException() {
        super();
        this.myEsterelParser = null;
    }
    /**
     * sets error messgage.
    * @param arg0 the text
    */
    public EsterelParserException(final String arg0) {
        super(arg0);
    }
    /**
     * sets error messgae and parser state.
     * @param arg0 the error message
     * @param aparser the parser state
     * @see kiel.fileInterface.esterel.EsterelParser
         */
    public EsterelParserException(
        final String arg0,
        final EsterelParser aparser) {
        this(arg0);
        this.myEsterelParser = aparser;
    }
    /**
     * @return Returns the myEsterelParser.
     */
    public final EsterelParser getMyEsterelParser() {
        return this.myEsterelParser;
    }
    /**
     * Shows a error pane.
     */
    public final void showMessageBox() {
      JOptionPane.showMessageDialog(null, this.getLocalizedMessage(), "Error",
                                    JOptionPane.ERROR_MESSAGE);
    }
}
```

C.1.4. EsterelParser

Auflistung C.4: Die Klasse `EsterelParser`

```
package kiel.fileInterface.esterel;

import java.util.List;
import kiel.fileInterface.esterel.esterel2estudio.AbortEsterelStatement;
import kiel.fileInterface.esterel.esterel2estudio.AssignEsterelStatement;
import kiel.fileInterface.esterel.esterel2estudio.AwaitEsterelStatement;
import kiel.fileInterface.esterel.esterel2estudio.DoUpToEsterelStatement;
import kiel.fileInterface.esterel.esterel2estudio.DoWatchingEsterelStatement;
import kiel.fileInterface.esterel.esterel2estudio.EmitEsterelStatement;
import kiel.fileInterface.esterel.esterel2estudio.ExitEsterelStatement;
import kiel.fileInterface.esterel.esterel2estudio.EsterelModule;
import kiel.fileInterface.esterel.esterel2estudio.EsterelStatement;
import kiel.fileInterface.esterel.esterel2estudio.EveryEsterelStatement;
import kiel.fileInterface.esterel.esterel2estudio.HaltEsterelStatement;
import kiel.fileInterface.esterel.esterel2estudio.IfEsterelStatement;
import kiel.fileInterface.esterel.esterel2estudio
.LocalSignalDeclarationEsterelStatement;
import kiel.fileInterface.esterel.esterel2estudio.LocalVariableEsterelStatement;
import kiel.fileInterface.esterel.esterel2estudio.LoopEachEsterelStatement;
import kiel.fileInterface.esterel.esterel2estudio.LoopEsterelStatement;
import kiel.fileInterface.esterel.esterel2estudio.NothingEsterelStatement;
import kiel.fileInterface.esterel.esterel2estudio.ParallelEsterelStatement;
import kiel.fileInterface.esterel.esterel2estudio.PauseEsterelStatement;
import kiel.fileInterface.esterel.esterel2estudio.PresentThenElse;
import kiel.fileInterface.esterel.esterel2estudio.ProcedureCallEsterelStatement;
import kiel.fileInterface.esterel.esterel2estudio.RepeatEsterelStatement;
import kiel.fileInterface.esterel.esterel2estudio.SequenceEsterelStatement;
import kiel.fileInterface.esterel.esterel2estudio.SuspendEsterelStatement;
import kiel.fileInterface.esterel.esterel2estudio.SustainEsterelStatement;
import kiel.fileInterface.esterel.esterel2estudio.TaskCallEsterelStatement;
import kiel.fileInterface.esterel.esterel2estudio.TrapEsterelStatement;
import kiel.util.DOMHelpers;
import org.jdom.Document;
import org.jdom.Element;

/**
 * <p>
 *  Parses an .exp file which was created by the cec.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2004
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.36 $ last modified $Date: 2006/02/07 15:39:35 $ <br>
 */
public class EsterelParser {
    /**
     * the actual parsed esterel module.
     */
    private int actualEsterModule = -1;

    /** The first free index in <code>anEsterelModul</code>. */
    private int addat = 0;

    /**
     * The index of the last parsed statement in <code>anEsterelModul</code>.
     */
    private int lastparsedStatement = 0;

    /** The number of statements in <code>anEsterelModul</code>. */
    private int statementCounter = 0;

    /** The actual parsed Element of a .exp file. */
    private Element theElement = null;

    /** The XPath search string which leads to the module in the .exp file. */
    private String theElementXPathSearchString =
        "//*[local-name(.)='Module']/*";

    /** A representation of an esterel module. */
    private EsterelModule[] theEsterelModules = null;

    /** The <code>org.jdom.Document</code> of a .exp file. */
    private Document xmldoc = null;

    /**
     * Simple constructor.
     */
    public EsterelParser() {
        this.xmldoc = null;
        this.theEsterelModules = null;
    } // public EsterelParser()

    /**
     * This constructor parses the Document with parseAstFile.
     *
     * @param anXmldoc
     *            the <code>org.jdom.Document</code> of an .exp(.xml) file
     * @throws EsterelParserException
     *             is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     * @see org.jdom.Document
     */
    public EsterelParser(final Document anXmldoc)
            throws EsterelParserException {
        this.parseExpFile(anXmldoc);
    } // public EsterelParser(final Document xmldoc)

    /**
```

```
     * Returns the <code>EsterelStatement</code> of the first statement.
     *  From the esterel module file which is parsed.
     * Sideeffects: The class variable <code>theElement</code> is set
     *
     * @return <code>EsterelStatement</code>
     * @throws EsterelParserException
     *           is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
     */
    public final EsterelStatement createProgam() throws EsterelParserException {
        this.theElementXPathSearchString += "/*[local-name(.)='Module']/*";
        List anElementList = DOMHelpers.getElements(
                this.theElementXPathSearchString, this.xmldoc);
        if (anElementList.size() > 0) {
            this.theElement = (Element) anElementList
                    .get(anElementList.size() - 1);
            return createStatement(this.theElement);
        }
        return null;
    } // public final EsterelStatement createProgam()

    /**
     * <p>
     * Returns the <code>EsterelStatement</code> of the actual parsed
     * <code>Element</code>.
     * </p>
     * <br>
     * Sideeffects: The class variable <code>theElement</code> is set
     *
     * @return <code>EsterelStatement</code>
     * @param anElement
     *          the <code>org.jdom.Element</code> which is parsed
     * @throws EsterelParserException
     *           is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
     * @see org.jdom.Element
     */
    public final EsterelStatement createStatement(final Element anElement)
            throws EsterelParserException {
        String toIdentifyStatement = anElement.getName();
        // System.out.println(toIdentifyStatement);
        this.theElement = anElement;
        EsterelStatement aReturnStatement = null;
        if (toIdentifyStatement == "ParallelStatementList") {
            aReturnStatement = new ParallelEsterelStatement(this);
        } else if (toIdentifyStatement == "Await") {
            aReturnStatement = new AwaitEsterelStatement(this);
        } else if (toIdentifyStatement == "Loop") {
            aReturnStatement = new LoopEsterelStatement(this);
        } else if (toIdentifyStatement == "StatementList") {
            aReturnStatement = new SequenceEsterelStatement(this);
        } else if (toIdentifyStatement == "Emit") {
            aReturnStatement = new EmitEsterelStatement(this);
        } else if (toIdentifyStatement == "Exit") {
            aReturnStatement = new ExitEsterelStatement(this);
        } else if (toIdentifyStatement == "Pause") {
            aReturnStatement = new PauseEsterelStatement(this);
        } else if (toIdentifyStatement == "Present") {
            aReturnStatement = new PresentThenElse(this);
        } else if (toIdentifyStatement == "Trap") {
            aReturnStatement = new TrapEsterelStatement(this);
        } else if (toIdentifyStatement == "Suspend") {
            aReturnStatement = new SuspendEsterelStatement(this);
        } else if (toIdentifyStatement == "Signal") {
            aReturnStatement = new LocalSignalDeclarationEsterelStatement(
                    this);
        } else if (toIdentifyStatement == "Nothing") {
            aReturnStatement = new NothingEsterelStatement(this);
        } else if (toIdentifyStatement == "Sustain") {
            aReturnStatement = new SustainEsterelStatement(this);
        } else if (toIdentifyStatement == "If") {
            aReturnStatement = new IfEsterelStatement(this);
        } else if (toIdentifyStatement == "Halt") {
            aReturnStatement = new HaltEsterelStatement(this);
        } else if (toIdentifyStatement == "Abort") {
            aReturnStatement = new AbortEsterelStatement(this);
        } else if (toIdentifyStatement == "LoopEach") {
            aReturnStatement = new LoopEachEsterelStatement(this);
        } else if (toIdentifyStatement == "Every") {
            aReturnStatement = new EveryEsterelStatement(this);
        } else if (toIdentifyStatement == "Repeat") {
            aReturnStatement = new RepeatEsterelStatement(this);
        } else if (toIdentifyStatement == "Var") {
            aReturnStatement = new LocalVariableEsterelStatement(this);
        } else if (toIdentifyStatement == "Assign") {
            aReturnStatement = new AssignEsterelStatement(this);
        } else if (toIdentifyStatement == "ProcedureCall") {
            aReturnStatement = new ProcedureCallEsterelStatement(this);
        } else if (toIdentifyStatement == "Run") {
            aReturnStatement = new NothingEsterelStatement(this);
        } else if (toIdentifyStatement == "Exec") {
            aReturnStatement = new TaskCallEsterelStatement(this);
        } else if (toIdentifyStatement == "TaskCall") {
            aReturnStatement = new TaskCallEsterelStatement(this);
        } else if (toIdentifyStatement == "DoWatching") {
            aReturnStatement = new DoWatchingEsterelStatement(this);
        } else if (toIdentifyStatement == "DoUpto") {
            aReturnStatement = new DoUpToEsterelStatement(this);
        } else {
            throw new EsterelParserException("Unknown Statement:"
                    + toIdentifyStatement, this);
        }
        if (aReturnStatement != null) {
            this.addat++;
        } else {
            throw new EsterelParserException("Unknown Statement:"
                    + toIdentifyStatement, this);
        }
        return aReturnStatement;
    }
```

```
    /**
     * Returns an index of the actual parsed esterel module. <br>
     * Sideeffects: none
     *
     * @return int
     */
    public final int getactualEsterModule() {
        return this.actualEsterModule;
    } // public final int getAddAt() {

    /**
     * Returns an index for the next added statement. <br>
     * Sideeffects: none
     *.
     * @return int
     */
    public final int getAddAt() {
        return this.addat;
    } // public final int getAddAt() {

    /**
     * <p>
     * Returns <code>theElement</code> of the Element which is parsed in that
     * moment.
     * </p>
     * <br>
     * Sideeffects: none
     *
     * @return org.jdom.Element
     * @see org.jdom.Element
     */
    public final Element getElement() {
        return this.theElement;
    } // public final Element getElement()

    /**
     * Returns the actual <code>EsterelModule</code>.<br>
     * Sideeffects: none
     *
     * @param i
     *          the index of the Module
     * @return <code>EsterelModule</code>
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    public final EsterelModule getEsterelModule(final int i) {
        return this.theEsterelModules[i];
    } // public final EsterelModule getEsterelModule() {

    /**
     * Returns the actual indexnumber from last parsed statement. <br>
     * Sideeffects: none
     *
     * @return int
     */
    public final int getLastParsedStatement() {
        return this.lastparsedStatement;
    } // public final int getLastParsedStatement() {

    /**
     * Returns the actual number of statements. <br>
     * Sideeffects: none
     *
     * @return int
     */
    public final int getStatementCounter() {
        return this.statementCounter;
    } // public final int getStatementCounter() {

    /**
     * <p>
     * Returns <code>theElementXPathSearchString</code> of the Element which
     * is parsed in that moment.
     * </p>
     * <br>
     * Sideeffects: none
     *
     * @return String
     */
    public final String gettheElementXPathSearchString() {
        return this.theElementXPathSearchString;
    } // public final String getXPathSearchString() {

    /**
     * Returns the actual <code>EsterelModule</code>.<br>
     * Sideeffects: none
     *
     * @return <code>EsterelModule</code>
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    public final EsterelModule[] gettheEsterelModules() {
        return this.theEsterelModules;
    } // public final EsterelModule getEsterelModule() {

    /**
     * <p>
     * Returns <code>org.jdom.Document</code> of the .exp file.
     * </p>
     * <br>
     * Sideeffects: none
     *
     * @return <code>org.jdom.Element</code>
     */
    public final Document getXMLDocument() {
        return this.xmldoc;
    } // public Document getXMLDocument()

    /**
     * Increments the class variable <code>statementcounter</code> by 1. <br>
     * Sideeffects: The class variable <code>statementcounter</code> is
     * increased by 1.<br>
     */
```

```
    public final void incStatementCounter() {
        this.statementCounter++;
330 } // public final void incStatementCounter()

    /**
     * <p>
     * Increments the class variable <code>statementcounter</code> by
     * <code>inc</code>.
     * </p>
     * <br>
     * Sideeffects: The class variable <code>statementcounter</code> is
     * increased by <code>inc</code>.<br>
340  *
     * @param inc
     *        Add <code>inc</code> to <code>statementcounter</code>.
     */
    public final void incStatementCounter(final int inc) {
        this.statementCounter += inc;
    } // public final void incStatementCounter(final int inc)

    /**
     * <p>
350  * Parses a <code>org.jdom.Document</code> into the class variable
     * <code>theEsterelModules</code>.
     * </p>
     * <br>
     * <br>
     * Sideeffects: The class variable <code>anEsterelModule</code> is set.
     * <br>
     * The class variable <code>xmldoc</code> is set with the paramenter. <br>
     * The class variable <code>theElement</code> is used. <br>
     *
360  * @param anXmldoc
     *        the Document of an .exp(.xml) file
     * @throws EsterelParserException
     *         is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    public final void parseExpFile(final Document anXmldoc)
            throws EsterelParserException {
        this.xmldoc = anXmldoc;
370     this.theElement = null;
        List allModules = DOMHelpers.getElements(
            "//*[local-name(.)='ModuleSymbol']", anXmldoc);
        if (allModules.size() < 0) {
            if (allModules.size() < 0) {
                throw new EsterelParserException("No module");
            }
        }
        this.theEsterelModules = new EsterelModule[allModules.size()];
        // set the names of the modules
380     for (this.actualEsterModule = 0;
                this.actualEsterModule < this.theEsterelModules.length;
                this.actualEsterModule++) {
            this.theElementXPathSearchString = "//*[@id='"
                    + ((Element) allModules.get(this.actualEsterModule))
                            .getAttributeValue("id") + "']";
            this.theEsterelModules[this.actualEsterModule] = new EsterelModule(
                    this.theElementXPathSearchString);
            this.theEsterelModules[this.getactualEsterModule()]
                    .parseEsterelModule(this);
        } // parse the modules
390 for (this.actualEsterModule = 0;
        this.actualEsterModule < this.theEsterelModules.length;
        this.actualEsterModule++) {
            this.lastparsedStatement = 0;
            this.addat = 0;
            this.statementCounter = 0;
            this.theElement = null;
            this.theElement = null;
            this.theElementXPathSearchString = "//*[@id='"
                    + ((Element) allModules.get(this.actualEsterModule))
                            .getAttributeValue("id") + "']";
400         this.theEsterelModules[this.actualEsterModule]
                    .addStatementToModule(this.addat, this.createProgam());
            while (this.lastparsedStatement < this.statementCounter) {
                // this.anEsterelModule.printProgram();
                this.theElementXPathSearchString =
                        ((EsterelStatement) this.theEsterelModules[this
                                .actualEsterModule]
                                .getEsterelProgram().get(this.lastparsedStatement))
                                .getEsterelStatementName();
410             ((EsterelStatement)
                        this.theEsterelModules[this.actualEsterModule]
                        .getEsterelProgram().get(this.lastparsedStatement))
                        .setSubStatments(this);
                this.lastparsedStatement++;
            } // while
        }
    }
420 /**
     * Returns a string.
     *
     * @return returns the string representation of a parser
     */
    public final String toString() {
        String result = "";
        for (int i = 0; i < this.theEsterelModules.length; i++) {
            result += this.theEsterelModules[i].toString() + "\n\n";
        }
        return result;
    }
}
```

C.1.5. EsterelProperties

Auflistung C.5: Die Klasse `EsterelProperties`

```
/*
 * $Author: lku $ $Date: 2006/02/06 18:35:28 $
 */
package kiel.fileInterface.esterel;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;

/**
 * <p>
 *  Used to load and store user defineable properties.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars K&uumlhl </a>
 * @version $Revision: 1.19 $
 */
public final class EsterelProperties {
    /**
     * The propertie key for automatic optimization .
     */
    private static final String AUTOMATICOPTIMIZATION =
        "AutomaticOptimizationOfEsterelStateCharts";

    /**
     * The propertie key for cygwins bin path under cygwin.
     */
    private static final String BINPATHINCYGWIN = "cygWinBinPath";

    /**
     * The propertie key for cygwin directory under windows.
     */
    private static final String CYGWINDIR = "cygWinDir";
    /**
     * The property key for debug mode.
     */
    private static final String DEBUGMODE = "DebugMode";

    /**
     * The default ressource for the properties.
     */
    private static final String DEFAULTRESOURCE = "esterel.properties";

    /**
     * The propertie key for number of states for the automatic
     * optimization.
     */
    private static final String MINMALNUMBEROFSTATESFOROPTIMIZATIONSTART
    = "NumberOfStatesToStartWithOptimization";

    /**
     * The propertie key for the cec logfile.
     */
    private static final String NAMEOFLOGFILE = "cecLogfileName";

    /**
     * These are the internal properties.
     */
    private static Properties properties;

    /**
     * This is the key for user specific file.
     */
    private static final String PROPERTIESFILE = System.getProperty("user.home")
           + File.separator
           + ".kiel"
           + File.separator
           + "esterel.properties";

    /**
     * path to the cec under Linux.
     */
    private static final String RELATIVLINUXPATHTOCEC = "cecLinuxPath";

    /**
     * path to the cec under windows.
     */
    private static final String RELATIVWINPATHTOCEC = "cecWinPath";

    /**
     * @return if automatic optimization is on means true.
     */
    public static boolean getAutomaticOptimization() {
        return properties.getProperty(AUTOMATICOPTIMIZATION)
                .equalsIgnoreCase("true");
    }
    /**
     * @return bin path under cygwin.
     */
    public static String getBinPathCygWin() {
        return properties.getProperty(BINPATHINCYGWIN);
    }

    /**
```

```
     * @return the path to the cec
     */
    public static String getCecPath() {
        if (System.getProperty("os.name")
                .toLowerCase().compareTo("linux") == 0) {
            return properties.getProperty(RELATIVLINUXPATHTOCEC);
        }
        return properties.getProperty(RELATIVWINPATHTOCEC);
    }

    /**
     * @return the cygwinpath.
     */
    public static String getCygwinPath() {
        return properties.getProperty(CYGWINDIR);
    }
    /**
     * @return true, if debugmode is true.
     */
    public static boolean getDebugMode() {
        return properties.getProperty(DEBUGMODE)
                .equalsIgnoreCase("true");
    }

    /**
     * @return name of ceclogfile.
     */
    public static String getNameOfLogfile() {
        return properties.getProperty(NAMEOFLOGFILE);
    }

    /**
     * @return the number of states needed for optimization.
     */
    public static int getNrOfStatesForAutomaticOptimization() {
        return Integer.parseInt(properties.getProperty(
            MINMALNUMBEROFSTATESFOROPTIMIZATIONSTART));
    }

    /**
     * Copies properties to user specific file.
     *
     * @return true, if copy was successful
     */
    private static boolean copyDefaults() {
        InputStream is;
        FileWriter fw;
        boolean success = true;
        try {
            is = EsterelProperties.class.getResourceAsStream(DEFAULTRESOURCE);
            fw = new FileWriter(
                    PROPERTIESFILE);
            int c = is.read();
            while (c >= 0) {
                fw.write(c);
                c = is.read();
            }
            fw.flush();
            is.close();
            fw.close();
        } catch (IOException storeExeception) {
            success = false;
        }
        return success;
    }

    /**
     * Loads the defaults form resource file.
     *
     * @return the default values
     */
    private static Properties getDefaults() {
        Properties defaults = new Properties();
        try {
            defaults.load(EsterelProperties.class.
                getResourceAsStream(DEFAULTRESOURCE));
        } catch (IOException e) {
            defaults.clear();
        }
        return defaults;
    }

    /**
     * Loads properties from user specific file.
     *
     * @return true, if loading was successful
     */
    private static boolean loadProperties() {
        boolean success = true;
        try {
            properties.load(new FileInputStream(
                    PROPERTIESFILE));
        } catch (IOException loadException) {
            success = false;
        }
        return success;
    }

    /**
     * This method will load the properties from local file. If this file
     * does not exist, the defaults are written to local file.
     *
     * @return true, if loading was successful
     */
    protected static boolean load() {
        boolean success;
        properties = new Properties(
                getDefaults());
        if (!loadProperties()) {
            success = copyDefaults();
        } else {
            success = true;
```

```
    }
    return success;
  }

  /**
   * Reloads the properties file.
   */
  protected static void reload() {
    loadProperties();
  }

  }

  /**
   * This is the construtor.
   */
  private EsterelProperties() {
  }

}
```

C.2. Paket `kiel.fileInterface.esterel.esterel2estudio`

Das Paket `kiel.fileInterface.esterel.esterel2estudio` enthält die Klassen zur Implementation der Transformationsregeln aus Kapitel 3. Zusätzlich werden die Klassen als UML-Darstellung angegeben.

C.2.1. AbortEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.1 dargestellt.

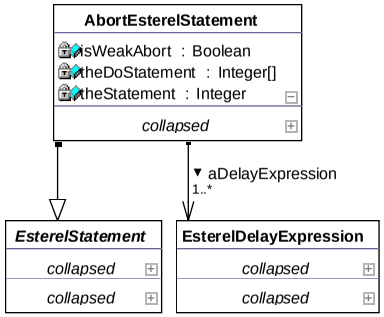


Abbildung C.1.: Klassediagramm `AbortEsterelStatement`

Auflistung C.6: Die Klasse `AbortEsterelStatement`

```
package kiel.fileInterface.esterel.esterel2estudio;

import java.util.ArrayList;
import java.util.List;

import kiel.dataStructure.CompositeState;
import kiel.dataStructure.CompoundLabel;
import kiel.dataStructure.FinalORState;
import kiel.dataStructure.FinalSimpleState;
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.Node;
import kiel.dataStructure.ORState;
import kiel.dataStructure.SimpleState;
import kiel.dataStructure.State;
import kiel.dataStructure.StringLabel;
import kiel.dataStructure.Transition;
import kiel.dataStructure.WeakAbortion;
import kiel.dataStructure.eventexp.DelayExpression;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StateChartHelpers;
import org.jdom.Element;

/**
 * <p>
 * Implements the esterel <code>abort</code> statement.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.40 $ last modified $Date: 2006/02/06 18:35:28 $
 *              <br>
 */
public class AbortEsterelStatement
       extends EsterelStatement {
   /** The value of the Expressions. */
   private EsterelDelayExpression[] aDelayExpression;
   /**
    * If it is a weak abort or a strong abort.
    */
   private boolean isWeakAbort;
```

```
/**
 * The signal to halt an abort handler excecution.
 */
private EsterelSignal myHaltSignal = null;

/**
 * the do statements.
 */
private int[] theDoStatement;

/**
 * the statment which to abort.
 */
private int theStatement;

/**
 * the standart constructor.
 */
public AbortEsterelStatement() {
    super();
}

/**
 * Parses a Document and set the class varaibles. <br>
 * It calls also the super constuctor <code>EsterelStatement
 * (EsterelParser theparser)</code>.
 * <br>
 * Calls the <code>parseEsterelStatement</code> methode <br>
 * Sideeffects: Changes <code>EsterelParser</code> class variables
 *
 * @param theparser
 *                an intace of a EsterelParser
 * @throws EsterelParserException
 *                is only delivered
 * @see kiel.fileInterface.esterel.EsterelParserException
 * @see #parseEsterelStatement(EsterelParser)
 * @see EsterelStatement#EsterelStatement(EsterelParser)
 * @see kiel.fileInterface.esterel.EsterelParser
 */
public AbortEsterelStatement(final EsterelParser theparser)
        throws EsterelParserException {
    super(
            theparser);
    if (theparser != null) {
        this.parseEsterelStatement(theparser);
        this.myHaltSignal =
            theparser.getEsterelModule(
                theparser.getactualEsterModule())
        .addNewSignal();
        this.myHaltSignal.setTrap(true);
        this.myHaltSignal.setMySignalIdentifier("halt"
                + this.getEsterelStatementName());
        this.myHaltSignal.createNewIdentifierName(
            theparser.getEsterelModule(theparser.getactualEsterModule()));
    } else {
        throw new EsterelParserException(
                "No Parser");
    }
}

/**
 * Returns a state representation of the esterel statement.
 *
 * @param anEsterelModule
 *                the esterel module which becomes a KIEL Statechart
 * @param isFinalState
 *                is true if the new state has to be a final state
 * @param theLocalEvents
 *                Stack with the local events
 * @param theLocalVariables
 *                Stack with all local variables
 * @param theTreesTrapSignals
 *                Stack with all tree trap signals.
 * @throws Esterel2EstudioException
 *                if the convertion to Kiel is not working. <br>
 *                Sideeffects: none
 * @return a State
 * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
 * @see kiel.dataStructure.State
 */
public final kiel.dataStructure.State convertToKiel(
       final EsterelModule anEsterelModule,
       final boolean isFinalState,
       final ArrayList theLocalEvents,
       final ArrayList theLocalVariables,
       final ArrayList theTreesTrapSignals)
       throws Esterel2EstudioException {
    int offset = 1;
    InitialArc aInitialArc = new InitialArc();
    InitialState anInitialState = new InitialState();
    FinalSimpleState theNormalEnd = new FinalSimpleState();
    SimpleState theTrapState = new SimpleState();
    CompositeState aState = null;
    if (isFinalState) {
        aState =
            new FinalORState(this.getEsterelStatementName() + "state");
    } else {
        aState = new ORState(this.getEsterelStatementName() + "state");
    }
    State theAbortedState =
        ((EsterelStatement) (anEsterelModule.getEsterelProgram()
                .get(this.theStatement)))
                .convertToKiel(anEsterelModule,
        false,
        theLocalEvents,
        theLocalVariables,
        theTreesTrapSignals);
    if (this.isWeakAbort
            && theTreesTrapSignals.size() > 0) {
        aState.addLocalEvent(this.myHaltSignal
                .convertToKiel(anEsterelModule));
        int oldsize = theLocalEvents.size();
```

```
    theLocalEvents.addAll(aState.getLocalEvents());
    theTreesTrapSignals.add(this);
    theAbortedState = ((EsterelStatement) anEsterelModule
            .getEsterelProgram()
            .get(this.theStatement))
            .convertToKiel(anEsterelModule,
        false,
        theLocalEvents,
        theLocalVariables,
        theTreesTrapSignals);
    theTreesTrapSignals.remove(theTreesTrapSignals.size() - 1);
    WeakAbortion aWA = StateChartHelpers.createWA(theAbortedState,
        offset,
        theTrapState);
    offset++;
    String aLabel = this.myHaltSignal.getSignalIdentifier();
    DelayExpression aDE = new DelayExpression();
    aDE.setEventExpression(StateChartHelpers
            .getBE(anEsterelModule.getEsterelStateChart(),
        theLocalVariables,
        theLocalEvents,
        aLabel));
    aDE.setImmediate(true);
    if (aDE.getEventExpression() == null) {
        aWA.setLabel(new StringLabel(
                "# "
                        + aLabel));
    } else {
        CompoundLabel aCL = new CompoundLabel();
        aCL.setTrigger(aDE);
        aWA.setLabel(aCL);
    }
    while (theLocalEvents.size() != oldsize) {
        theLocalEvents.remove(theLocalEvents.size() - 1);
    }
}
State[] theEndStates = new State[this.aDelayExpression.length];
for (int theCaseIndex = 0;
theCaseIndex < this.aDelayExpression.length;
theCaseIndex++) {
    if (this.theDoStatement[theCaseIndex] == -1) {
        theEndStates[theCaseIndex] = new FinalSimpleState();
    } else {
        theEndStates[theCaseIndex] =
            ((EsterelStatement)
                    (anEsterelModule.getEsterelProgram()
                            .get(this.theDoStatement[theCaseIndex])))
                            .convertToKiel(anEsterelModule,
            true,
            theLocalEvents,
            theLocalVariables,
            theTreesTrapSignals);
    }
    Transition aAwaitTransition = null;
    if (this.isWeakAbort) {
        aAwaitTransition = StateChartHelpers.createWA(theAbortedState,
                theCaseIndex
                        + offset,
                theEndStates[theCaseIndex]);
            aAwaitTransition
            .setLabel(
                    this.aDelayExpression[theCaseIndex]
                                          .convertToKiel(anEsterelModule,
                theLocalEvents,
                theLocalVariables,
                null));
     } else {
         aAwaitTransition = StateChartHelpers.createSA(theAbortedState,
                 theCaseIndex
                         + offset,
                 theEndStates[theCaseIndex]);
             aAwaitTransition
             .setLabel(
                     this.aDelayExpression[theCaseIndex]
                                           .convertToKiel(anEsterelModule,
                 theLocalEvents,
                 theLocalVariables,
                 null));
      }
    }
    aInitialArc.setSource(anInitialState);
    aInitialArc.setTarget(theAbortedState);
    // if theAbortedState is not aborted
    //then the normal termination is used
    StateChartHelpers.createNT(theAbortedState,
        this.theDoStatement.length + 1,
        theNormalEnd);
    int three = 1 + 1 + 1; // + 3 States
    if (this.isWeakAbort && theTreesTrapSignals.size() > 0) {
        three++;
    }
    Node[] theNodes = new Node[theEndStates.length
            + three];
    for (int i = 0; i < theNodes.length
            - three; i++) {
        theNodes[i] = theEndStates[i];
    }
    if (this.isWeakAbort && theTreesTrapSignals.size() > 0) {
        theNodes[theNodes.length
                - three
                + 1] = theTrapState;
    }
    theNodes[theNodes.length
            - three] = theNormalEnd;
    theNodes[theNodes.length - 2] = anInitialState;
    theNodes[theNodes.length - 1] = theAbortedState;
    for (int i = 0; i < theNodes.length; i++) {
        aState.addSubnode(theNodes[i]);
        theNodes[i].setParent(aState);
    }
    return aState;
}
```

```
/**
 * @return Returns the myHaltSignal.
 */
public final EsterelSignal getMyHaltSignal() {
    return this.myHaltSignal;
}

/**
 * Implemtents the abstrac methode from <code>EsterelStatement</code>
 * Parses an esterel statment in an <code>org.jdom.Document</code>
 * representation of an .exp file if you have not called the
 * constructor <code> EsterelStatement(EsterelParser theparser)</code>
 * use the methode <code>preParseEsterelStatement</code> before
 * <code>parseEsterelStatement</code>.<br>
 * Sideeffects: the class variables are set Changes
 * <code>EsterelParser</code> class variables
 *
 * @param theparser
 *                an instance of <code>EsterelParser</code>
 * @throws EsterelParserException
 *                is only delivered
 * @see kiel.fileInterface.esterel.EsterelParserException
 * @see EsterelStatement#EsterelStatement(EsterelParser)
 * @see EsterelStatement#parseEsterelStatement(EsterelParser)
 * @see EsterelStatement#parseEsterelStatement(EsterelParser)
 * @see kiel.fileInterface.esterel.EsterelParser
 * @see org.jdom.Document
 */
public final void parseEsterelStatement(
       final EsterelParser theparser)
       throws EsterelParserException {
    List anElementList = DOMHelpers.getElements(theparser.getXMLDocument(),
        this.getXPathSearchString());
    theparser.incStatementCounter(); // the
                                                 // statement to
                                                 // abort
    if (((Element) anElementList.get(anElementList.size() - 1)).getName()
          .compareTo("BoolTrue") == 0) {
        this.isWeakAbort = true;
    } else {
        this.isWeakAbort = false;
    }
    this.aDelayExpression =
        new EsterelDelayExpression[anElementList.size() - 2];
    //2 may be wrong
    this.theDoStatement = new int[anElementList.size() - 2];
    // set default values
    for (int i = 0; i < anElementList.size() - 2; i++) {
        this.aDelayExpression[i] = null;
        this.theDoStatement[i] = -1;
    }
    int doCounter = 0;
    for (int theCaseIndex = 0;
    theCaseIndex < anElementList.size() - 2;
    theCaseIndex++) {
        if (((Element) anElementList.get(theCaseIndex)).getName()
               == "PredicatedStatement") {
            String aPredicatedStatementXPathSearchString =
                this.getXPathSearchString()
                    + "[local-name(.)='"
                    + ((Element) anElementList.get(theCaseIndex)).getName()
                    + "'][@id='"
                    + ((Element) anElementList.get(theCaseIndex))
                    .getAttributeValue("id")
                    + "']/*";
            List aPredicatedElementList =
                DOMHelpers.getElements(theparser.getXMLDocument(),
                aPredicatedStatementXPathSearchString);
            if (aPredicatedElementList.size() > 1) {
                doCounter++;
            }
            int index = aPredicatedElementList.size() - 1;
            // the DelayExpression is the last listelement
            String aSubPredicatedStatementXPathSearchString =
                "//*[local-name(.)='"
                    + ((Element) aPredicatedElementList.get(index))
                    .getName()
                    + "'][@id='"
                    + ((Element) aPredicatedElementList.get(index))
                    .getAttributeValue("id")
                    + "']/*";
            this.aDelayExpression[theCaseIndex] =
                new EsterelDelayExpression(
                    theparser,
                    aSubPredicatedStatementXPathSearchString,
                    theparser.getXMLDocument());
        } else {
            throw new EsterelParserException(
                    this.getEsterelStatementName()
                           + " : No predicated statement found",
                    theparser);
        } //else if size*/
    }
    theparser.incStatementCounter(doCounter);
} //public final void parseEsterelStatement(final EsterelParser
  // theparser)

/**
 * Fills the substatements in the esterel module. In this class it is
 * fills the do Statements if there are any <br>
 * Sideeffects: none
 *
 * @param theparser
 *                the actual parser
 * @throws EsterelParserException
 *                is only delivered
 * @see kiel.fileInterface.esterel.EsterelParserException
 * @see kiel.fileInterface.esterel.EsterelParser
 */
public final void setSubStatments(
       final EsterelParser theparser)
```

```
    throws EsterelParserException {
List anElementList = DOMHelpers.getElements(theparser.getXMLDocument(),
    this.getXPathSearchString());
  // the statement which may be aborted is before the last element
  if (anElementList.size() - 1 > 0) {
    this.theStatement = theparser.getAddAt();
    theparser.getEsterelModule(theparser.getactualEsterModule())
        .addStatementToModule(theparser.getAddAt(),
            theparser
            .createStatement(((Element)
                (anElementList.get(anElementList.size() - 2)))));
  }
  // first the do statements
  for (int theCaseIndex = 0;
theCaseIndex < anElementList.size() - 2;
theCaseIndex++) {
    if (((Element) anElementList.get(theCaseIndex)).getName()
        == "PredicatedStatement") {
      String aPredicatedStatementXPathSearchString =
          this.getXPathSearchString()
          + "[local-name(.)='"
          + ((Element) anElementList.get(theCaseIndex)).getName()
          + "'][@id='"
          + ((Element) anElementList.get(theCaseIndex))
          .getAttributeValue("id")
          + "']/*";
      List aPredicatedElementList =
          DOMHelpers.getElements(theparser.getXMLDocument(),
          aPredicatedStatementXPathSearchString);
      if (aPredicatedElementList.size() > 1) {
        this.theDoStatement[theCaseIndex] = theparser.getAddAt();
        theparser.getEsterelModule(theparser.getactualEsterModule())
            .addStatementToModule(theparser.getAddAt(),
                theparser
                .createStatement(((Element)
                    (aPredicatedElementList.get(0)))));
      } else {
        this.theDoStatement[theCaseIndex] = -1;
      }
    } //if
  } //for
}

/**
 * Implementns the abstrac methode from <code>EsterelStatement</code>.
 * Returns a string representation of an esterel statement.
 *
 * @param anEsterelModule
 *            an esterel modul representation
 *
 * @return a String representation of an esterel statment
 * @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
 * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
 */
public final String toString(
    final EsterelModule anEsterelModule) {
  String result = "";
  if (this.isWeakAbort) {
    result += "weak ";
  }
  result += this.getEsterelStatementName();
  result += "\n"
      + ((EsterelStatement)
          (anEsterelModule
              .getEsterelProgram()
              .get(this.theStatement)))
              .toString(anEsterelModule);
  result += "\n when ";
  if (this.aDelayExpression.length > 1) {
    for (int theCaseIndex = 0;
    theCaseIndex < this.aDelayExpression.length;
    theCaseIndex++) {
      result += "\n"
          + "     case "
          + this.aDelayExpression[theCaseIndex]
              .toString(anEsterelModule);
      if (this.theDoStatement[theCaseIndex] != -1) {
        result += " do "
            + ((EsterelStatement) anEsterelModule
                .getEsterelProgram()
                .get(this.theDoStatement[theCaseIndex]))
                .toString(anEsterelModule);
      }
    }
    result += "\n end "
        + this.getEsterelStatementName();
  } else {
    result += " "
        + this.aDelayExpression[0] + "\n";
  }
  return result;
}
}
```

C.2.2. AssignEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.2 dargestellt.

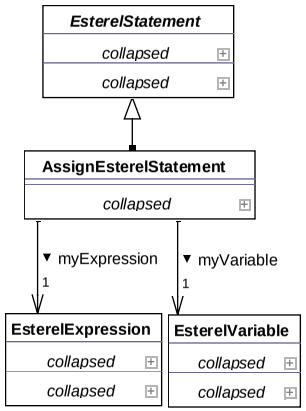


Abbildung C.2.: Klassediagramm `AssignEsterelStatement`

Auflistung C.7: Die Klasse `AssignEsterelStatement`

```
package kiel.fileInterface.esterel.esterel2estudio;
import java.util.ArrayList;
import java.util.List;
import kiel.dataStructure.CompoundLabel;
import kiel.dataStructure.FinalSimpleState;
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.Node;
import kiel.dataStructure.StringLabel;
import kiel.dataStructure.TransitionLabel;
import kiel.dataStructure.Variable;
import kiel.dataStructure.action.Actions;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StateChartHelpers;
import org.jdom.Element;
/**
 * <p> This is a subclass of <code>EsterelStatement</code>
 * classes implements the assign statement.</p>
 * <p>Copyright: Copyright (c) 2005</p>
 * <p>Company: Uni Kiel</p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
 * @version $Revision: 1.41 $ last modified $Date: 2006/02/17 20:00:22 $
 * <br>
 */
public class AssignEsterelStatement extends EsterelStatement {
    /**
     * the assigned Expression.
     */
    private EsterelExpression myExpression;
    /**
     * the variable.
     */
    private EsterelVariable myVariable;
    /**
     * simple constructor.
```

```
 */
public AssignEsterelStatement() {
    super();
    this.myExpression = null;
    this.myVariable = null;
}
/**
 * Parses a Document and set the class varaibles.
 * <br>It calls also the super constuctor<code>EsterelStatement
 * (EsterelParser theparser)</code> .
 * <br>Calls the <code>parseEsterelStatement</code> methode
 * <br>Sideeffects:
 * Changes <code>EsterelParser</code>class variables
 *@param theparser an intace of a EsterelParser
 * @throws EsterelParserException is only delivered
 * @see kiel.fileInterface.esterel.EsterelParserException
 * @see #parseEsterelStatement(EsterelParser)
 * @see EsterelStatement#EsterelStatement(EsterelParser)
 * @see kiel.fileInterface.esterel.EsterelParser
 */
public AssignEsterelStatement(final EsterelParser theparser)
    throws EsterelParserException {
    super(theparser);
    this.myExpression = null;
    this.myVariable = null;
    if (theparser != null) {
        this.parseEsterelStatement(theparser);
    } else {
        throw new EsterelParserException("No Parser");
    }
}
/**
 * Returns a state representation of the esterel statement.
 * @param anEsterelModule the esterel module which
 * becomes a KIEL Statechart
 *@param isFinalState is true if the new state has to be
 * a final state
 * @param theLocalEvents Stack with the local events
 * @param theLocalVariables Stack with all local variables
 * @param theTreesTrapSignals
 *          Stack with all tree trap signals.
 * @throws Esterel2EstudioException
 *          if the convertion to Kiel is not working. <br>
 *          Sideeffects: none
 *
 * @return a State
 * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
 * @see kiel.dataStructure.State
 */
public final kiel.dataStructure.State convertToKiel(
        final EsterelModule anEsterelModule,
        final boolean isFinalState,
        final ArrayList theLocalEvents,
        final ArrayList theLocalVariables,
        final ArrayList theTreesTrapSignals)
        throws Esterel2EstudioException {
    InitialArc aInitialArc = new InitialArc();
    InitialState anInitialState = new InitialState();
    FinalSimpleState theEndState = new FinalSimpleState();
    aInitialArc.setSource(anInitialState);
    aInitialArc.setTarget(theEndState);
    TransitionLabel theLabelofaTransition = null;
    Variable anVar = this.myVariable
    .convertToKiel(anEsterelModule, theLocalEvents, theLocalVariables);

    if (anVar != null && this.myExpression != null) {
        String aLabel = anVar.getName()
            + " := "
            + this.myExpression.toString(anEsterelModule);

            Actions theActions =
                StateChartHelpers
                .getActions(anEsterelModule.getEsterelStateChart(),
                        theLocalVariables,
                        theLocalEvents,
                        " " + aLabel);
            if (theActions == null) {
                theLabelofaTransition =
                    new StringLabel("/ " + aLabel);
            } else {
            theLabelofaTransition = new CompoundLabel();
            ((CompoundLabel) theLabelofaTransition).setEffect(theActions);
            }
    } else {
        throw new Esterel2EstudioException(
            this.getEsterelStatementName()
                + " : "
                + " no expression or variable");
    }
    aInitialArc.setLabel(theLabelofaTransition);
    if (isFinalState) {
        return StateChartHelpers.createFinalOR(
            this.getEsterelStatementName() + "state",
            new Node[] {anInitialState, theEndState });
    }
        return StateChartHelpers.createOR(
            this.getEsterelStatementName() + "state",
            new Node[] {anInitialState, theEndState });
}
/**
 *Implementents the abstrac methode from <code>EsterelStatement</code>
 * Parses an esterel statment in an <code>org.jdom.Document</code>
 * representation of an .exp file
 *  if you have not called the constructor
 * <code> EsterelStatement(EsterelParser theparser)</code>
 * use the methode <code>preParseEsterelStatement</code>
 * before <code>parseEsterelStatement</code>.
 *  <br>Sideeffects:
 * the class variables are set
 * Changes <code>EsterelParser</code>class variables
 * @param theparser an instance of <code>EsterelParser</code>
 * @throws EsterelParserException is only delivered
```

```
     * @see kiel.fileInterface.esterel.EsterelParserException
     * @see EsterelStatement#EsterelStatement(EsterelParser)
     * @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see kiel.fileInterface.esterel.EsterelParser
     * @see org.jdom.Document
     */
    public final void parseEsterelStatement(final EsterelParser theparser)
        throws EsterelParserException {
        List anElementList =
            DOMHelpers.getElements(
                theparser.getXMLDocument(),
                this.getXPathSearchString());
        if (anElementList.size() == 2) {
            this.myVariable =
                theparser
                .getEsterelModule(theparser
                    .getactualEsterModule())
                .getEsterelVariableByID(
                    ((Element) anElementList.get(0))
                    .getAttributeValue("id"));

            this.myExpression =
                new EsterelExpression(
                    theparser,
                    this.getXPathSearchString()
                        + "[local-name(.)='"
                        + ((Element) anElementList.get(1)).getName()
                        + "'][@id='"
                        + ((Element) anElementList.get(1)).getAttributeValue(
                            "id")
                        + "']/*",
                    theparser.getXMLDocument());
        } else {
            throw new EsterelParserException(
                this.getEsterelStatementName()
                    + " : "
                    + " not correct number of subelements");
        }
    }
    /**
     * Fills the substatements in the esterel module.
     * In this class it is fills the do Statements if there are any
     *<br> Sideeffects: none
     * @param theparser the actual parser
     * @throws EsterelParserException is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     *  @see kiel.fileInterface.esterel.EsterelParser
     */
    public final void setSubStatments(final EsterelParser theparser)
        throws EsterelParserException { }
    /**
     *Implementents the abstrac methode from <code>EsterelStatement</code>.
     * Returns a string representation of an esterel statement.
     * @param anEsterelModule an esterel modul representation
     * @return a String representation of an esterel statment
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
     *@see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    public final String toString(final EsterelModule anEsterelModule) {
        return this.myVariable
            .getVariableIdentifier()
            + " := "
            + this.myExpression.toString(anEsterelModule)
            + "\n";
    }
}
```

C.2.3. AwaitEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.3 dargestellt.

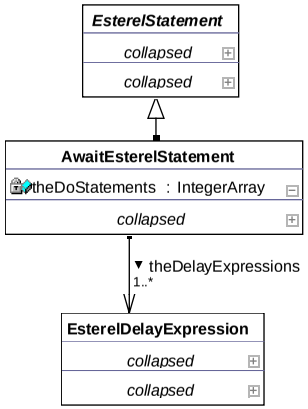


Abbildung C.3.: Klassediagramm `AwaitEsterelStatement`

Auflistung C.8: Die Klasse `AwaitEsterelStatement`

```
package kiel.fileInterface.esterel.esterel2estudio;
//
import java.util.ArrayList;
import java.util.List;
//
import kiel.dataStructure.FinalSimpleState;
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.Node;
import kiel.dataStructure.SimpleState;
import kiel.dataStructure.State;
import kiel.dataStructure.Transition;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StateChartHelpers;
//
import org.jdom.Element;
/**
 * <p> This is a subclass of <code>EsterelStatement</code>
 * classes implements the await statement.</p>
 * It implements await s,await s do p; await case s do p; await case s
 * <p>Copyright: Copyright (c) 2004</p>
 * <p>Company: Uni Kiel</p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
 * @version $Revision: 1.55 $ last modified $Date: 2006/02/06 18:35:28 $
 * <br>
 * $Log: AwaitEsterelStatement.java,v $
 * Revision 1.55 2006/02/06 18:35:28 lku
 * *** empty log message ***
 *
 * Revision 1.52 2005/11/10 12:26:25 lku
 * *** empty log message ***
 *
 * Revision 1.44 2005/10/04 11:33:28 lku
 * *** empty log message ***
 *
 * Revision 1.24 2005/05/12 01:37:57 lku
```

```
 * *** empty log message ***
 *
 * Revision 1.9 2005/01/14 16:42:26 lku
 * *** empty log message ***
 *
 * <br>
 * Revision 1.8 2005/01/12 15:00:00 lku
 * <br>
 * add javadoc + check for project manual code conventions
 * <br>
 *Revision 1.10
 *change aDelayExpression from String to EsterelDelayExpression
 *also major changes in set DelayExpression and convertToKiel
 *add doStatment for await x do p
 *add case
*/
public class AwaitEsterelStatement extends EsterelStatement {
    /** The value of the Expressions.*/
    private EsterelDelayExpression[] aDelayExpression;
    /**
     * the do statements.
     */
    private int[] theDoStatement;
    /**
     * Simple constructor.
     */
    public AwaitEsterelStatement() {
        super();
    }
    /**
    * Parses a Document and set the class varaibles.
    * <br>It calls also the super constuctor<code>EsterelStatement
    * (EsterelParser theparser)</code> .
    * <br>Calls the <code>parseEsterelStatement</code> methode
    * <br>Sideeffects:
    * Changes <code>EsterelParser</code>class variables
    *@param theparser an intace of a EsterelParser
    * @throws EsterelParserException is only delivered
    * @see kiel.fileInterface.esterel.EsterelParserException
    * @see #parseEsterelStatement(EsterelParser)
    * @see EsterelStatement#EsterelStatement(EsterelParser)
    * @see kiel.fileInterface.esterel.EsterelParser
    */
    public AwaitEsterelStatement(final EsterelParser theparser)
        throws EsterelParserException {
        super(theparser);
        if (theparser != null) {
            this.parseEsterelStatement(theparser);
        } else {
            throw new EsterelParserException("No Parser");
        }
    } //   public AwaitEsterelStatement(final EsterelParser theparser) {
    /**
     * Returns a state representation of the esterel statement.
     * @param anEsterelModule the esterel module which
     * becomes a KIEL Statechart
     *@param isFinalState is true if the new state has to be
     * a final state
     * @param theLocalEvents Stack with the local events
     * @param theLocalVariables Stack with all local variables
     * @param theTreesTrapSignals
     *                Stack with all tree trap signals.
     * @throws Esterel2EstudioException
     *                if the convertion to Kiel is not working. <br>
     *                Sideeffects: none
     * @return a State
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     * @see kiel.dataStructure.State
     */
    public final kiel.dataStructure.State convertToKiel(
            final EsterelModule anEsterelModule,
            final boolean isFinalState,
            final ArrayList theLocalEvents,
            final ArrayList theLocalVariables,
            final ArrayList theTreesTrapSignals)
            throws Esterel2EstudioException {
        InitialArc aInitialArc = new InitialArc();
        InitialState anInitialState = new InitialState();
        SimpleState theStartState = new SimpleState();
        State[] theEndStates = new State[this.aDelayExpression.length];
        for (int theCaseIndex = 0;
            theCaseIndex < this.aDelayExpression.length;
            theCaseIndex++) {
            if (this.theDoStatement[theCaseIndex] == -1) {
                theEndStates[theCaseIndex] = new FinalSimpleState();
            } else {
                theEndStates[theCaseIndex] =
                    (
                        (EsterelStatement)
                            (
                                anEsterelModule.getEsterelProgram().get(
                                    this
                                        .theDoStatement[theCaseIndex]))
                                        .convertToKiel(
                        anEsterelModule,
                        true,
                        theLocalEvents,
                        theLocalVariables,
                        theTreesTrapSignals);
            }
            Transition aAwaitTransition =
                StateChartHelpers.createSA(
                    theStartState,
                    theCaseIndex + 1,
                    theEndStates[theCaseIndex]);
            aAwaitTransition.setLabel(
                this.aDelayExpression[theCaseIndex].convertToKiel(
                    anEsterelModule,
                    theLocalEvents,
                    theLocalVariables,
                    null));
            /*StateChartHelpers.setDelayExpression(
```

```
            aAwaitTransition,
            aDelayExpression[theCaseIndex].convertToKiel(
                anEsterelModule,
                theLocalEvents));*/
        }
        aInitialArc.setSource(anInitialState);
        aInitialArc.setTarget(theStartState);
        Node[] theNodes = new Node[theEndStates.length + 2];
        for (int i = 0; i < theNodes.length - 2; i++) {
            theNodes[i] = theEndStates[i];
        }
        theNodes[theNodes.length - 2] = anInitialState;
        theNodes[theNodes.length - 1] = theStartState;
        if (isFinalState) {
            return StateChartHelpers.createFinalOR(
                this.getEsterelStatementName() + "state",
                theNodes);
        }
            return StateChartHelpers.createOR(
                this.getEsterelStatementName() + "state",
                theNodes);

    }
    /**
     *Implemtents the abstrac methode from <code>EsterelStatement</code>
     * Parses an esterel statment in an <code>org.jdom.Document</code>
     * representation of an .exp file
     *  if you have not called the constructor
     * <code> EsterelStatement(EsterelParser theparser)</code>
     * use the methode <code>preParseEsterelStatement</code>
     * before <code>parseEsterelStatement</code>.
      *  <br>Sideeffects:
     * the class variables are set
     * Changes <code>EsterelParser</code>class variables
     * @param theparser an instance of <code>EsterelParser</code>
    * @throws EsterelParserException is only delivered
    * @see kiel.fileInterface.esterel.EsterelParserException
     * @see EsterelStatement#EsterelStatement(EsterelParser)
     *  @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see kiel.fileInterface.esterel.EsterelParser
     * @see org.jdom.Document
     */
    public final void parseEsterelStatement(final EsterelParser theparser)
        throws EsterelParserException {
        List anElementList =
            DOMHelpers.getElements(
                theparser.getXMLDocument(),
                this.getXPathSearchString());
        this.aDelayExpression =
            new EsterelDelayExpression[anElementList.size()];
        this.theDoStatement = new int[anElementList.size()];
        // set default values
        for (int i = 0; i < anElementList.size(); i++) {
            this.aDelayExpression[i] = null;
            this.theDoStatement[i] = -1;
        }
        int doCounter = 0;
        for (int theCaseIndex = 0;
            theCaseIndex < anElementList.size();
            theCaseIndex++) {
            if (((Element) anElementList.get(theCaseIndex)).getName()
                == "PredicatedStatement") {
                String aPredicatedStatementXPathSearchString =
                    this.getXPathSearchString()
                        + "[local-name(.)='"
                        + ((Element) anElementList.get(theCaseIndex)).getName()
                        + "'][@id='"
                        + (
                            (Element) anElementList.get(
                                theCaseIndex)).getAttributeValue(
                            "id")
                        + "']/*";
                List aPredicatedElementList =
                    DOMHelpers.getElements(
                        theparser.getXMLDocument(),
                        aPredicatedStatementXPathSearchString);
                if (aPredicatedElementList.size() > 1) {
                    doCounter++;
                }
                int index = aPredicatedElementList.size() - 1;
                // the DelayExpression is the last listelement
                this.aDelayExpression[theCaseIndex] =
                    new EsterelDelayExpression(
                        theparser,
                        "//*[local-name(.)='"
                            + ((Element) aPredicatedElementList.get(index))
                                .getName()
                            + "'][@id='"
                            + (
                                (Element) aPredicatedElementList.get(
                                    index)).getAttributeValue(
                                "id")
                            + "']/*",
                        theparser.getXMLDocument());
            } else {
                throw new EsterelParserException(
                    this.getEsterelStatementName()
                        + " : No predicated statement found",
                    theparser);
            } //else if size*/
        }
        theparser.incStatementCounter(doCounter);
    } //public final void parseEsterelStatement(final EsterelParser theparser)
    //  final EsterelModule anEsterelModule)
    /**
    * Fills the substatements in the esterel module.
    * In this class it is fills the do Statements if there are any
    *<br> Sideeffects: none
    * @param theparser the actual parser
    * @throws EsterelParserException is only delivered
    * @see kiel.fileInterface.esterel.EsterelParserException
```

```
     * @see kiel.fileInterface.esterel.EsterelParser
     */
    public final void setSubStatments(final EsterelParser theparser)
        throws EsterelParserException {
        List anElementList =
            DOMHelpers.getElements(
                theparser.getXMLDocument(),
                this.getXPathSearchString());
        for (int theCaseIndex = 0;
            theCaseIndex < anElementList.size();
            theCaseIndex++) {
            if (((Element) anElementList.get(theCaseIndex)).getName()
                == "PredicatedStatement") {
                String aPredicatedStatementXPathSearchString =
                    this.getXPathSearchString()
                        + "[local-name(.)='"
                        + ((Element) anElementList.get(theCaseIndex)).getName()
                        + "'][@id='"
                        + (
                            (Element) anElementList.get(
                                theCaseIndex)).getAttributeValue(
                            "id")
                        + "']/*";
                List aPredicatedElementList =
                    DOMHelpers.getElements(
                        theparser.getXMLDocument(),
                        aPredicatedStatementXPathSearchString);
                if (aPredicatedElementList.size() > 1) {
                    this.theDoStatement[theCaseIndex] = theparser.getAddAt();
                    theparser.getEsterelModule(
                        theparser.getactualEsterModule())
                        .addStatementToModule(
                        theparser.getAddAt(),
                        theparser.createStatement(
                            ((Element) (aPredicatedElementList.get(0)))));
                } else {
                    this.theDoStatement[theCaseIndex] = -1;
                }
                } //if
            } //for
        } //set
        /**
         *Implemtents the abstrac methode from <code>EsterelStatement</code>.
        * Returns a string representation of an esterel statement.
        * @param anEsterelModule an esterel modul representation
        * @return a String representation of an esterel statment
        * @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
        *@see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
        */
        public final String toString(final EsterelModule anEsterelModule) {
            String result = this.getEsterelStatementName();
            if (this.aDelayExpression.length > 1) {
                for (int theCaseIndex = 0;
                    theCaseIndex < this.aDelayExpression.length;
                    theCaseIndex++) {
                    result += "\n"
                        + "     case "
                        + this.aDelayExpression[theCaseIndex]
                                .toString(anEsterelModule);
                    if (this.theDoStatement[theCaseIndex] != -1) {
                        result += " do "
                            + (
                                (EsterelStatement) anEsterelModule
                                    .getEsterelProgram()
                                    .get(
                                    this.theDoStatement[theCaseIndex])).toString(
                                anEsterelModule);
                    }
                }
                result += "\n end " + this.getEsterelStatementName();
            } else {
                result += " " + this.aDelayExpression[0] + "\n";
            }
            return result;
        }
    } //public class AwaitEsterelStatement
```

C.2.4. DoUpToEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.4 dargestellt.

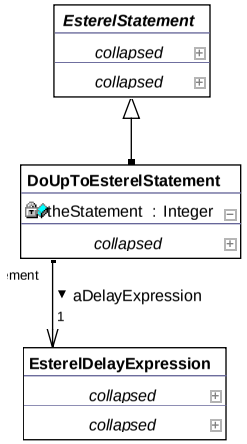


Abbildung C.4.: Klassediagramm `DoUpToEsterelStatement`

Auflistung C.9: Die Klasse `DoUpToEsterelStatement`

```
package kiel.fileInterface.esterel.esterel2estudio;
import java.util.ArrayList;
import java.util.List;
//
import kiel.dataStructure.FinalSimpleState;
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.Node;
import kiel.dataStructure.State;
import kiel.dataStructure.Transition;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StateChartHelpers;
//
import org.jdom.Element;

/**
 * <p>Implements the esterel <code>DoUpTo</code> statement.</p>
 * <p>Copyright: Copyright (c) 2005</p>
 * <p>Company: Uni Kiel</p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
 * @version $Revision: 1.35 $ last modified $Date: 2006/02/06 18:35:28 $
 * <br>
 */
public class DoUpToEsterelStatement extends EsterelStatement {
    /** The value of the Expression.*/
    private EsterelDelayExpression aDelayExpression;
    /**
     * the statment which to DoUpTo.
     */
    private int theStatement;
    /**
     * the standart constructor.
```

```
     *
     */
    public DoUpToEsterelStatement() {
        super();
    }
    /**
    * Parses a Document and set the class varaibles.
    * <br>It calls also the super constuctor<code>EsterelStatement
    * (EsterelParser theparser)</code> .
    * <br>Calls the <code>parseEsterelStatement</code> methode
    * <br>Sideeffects:
    * Changes <code>EsterelParser</code>class variables
    *@param theparser an intace of a EsterelParser
    * @throws EsterelParserException is only delivered
    * @see kiel.fileInterface.esterel.EsterelParserException
    * @see #parseEsterelStatement(EsterelParser)
    * @see EsterelStatement#EsterelStatement(EsterelParser)
    * @see kiel.fileInterface.esterel.EsterelParser
    */
    public DoUpToEsterelStatement(final EsterelParser theparser)
        throws EsterelParserException {
        super(theparser);
        this.theStatement = -1;
        if (theparser != null) {
            this.parseEsterelStatement(theparser);
        } else {
            throw new EsterelParserException("No Parser");
        }
    }
    /**
     * Returns a state representation of the esterel statement.
     * @param anEsterelModule the esterel module which
     * becomes a KIEL Statechart
     *@param isFinalState is true if the new state has to be
     * a final state
     * @param theLocalEvents Stack with the local events
     * @param theLocalVariables Stack with all local variables
     * @param theTreesTrapSignals
     *                  Stack with all tree trap signals.
     * @throws Esterel2EstudioException
     *                  if the convertion to Kiel is not working. <br>
     *                  Sideeffects: none
     * @return a State
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     * @see kiel.dataStructure.State
     */
    public final kiel.dataStructure.State convertToKiel(
           final EsterelModule anEsterelModule,
           final boolean isFinalState,
           final ArrayList theLocalEvents,
           final ArrayList theLocalVariables,
           final ArrayList theTreesTrapSignals)
           throws Esterel2EstudioException {
        InitialArc aInitialArc = new InitialArc();
        InitialState anInitialState = new InitialState();
        FinalSimpleState theEndState = new FinalSimpleState();
        State theDoUpToedState =
            (
                (EsterelStatement)
                   (
                       anEsterelModule.getEsterelProgram().get(
                           this.theStatement))).convertToKiel(
                anEsterelModule,
                false,
                theLocalEvents,
                theLocalVariables,
                theTreesTrapSignals);
        Transition aSATransition =
            StateChartHelpers.createSA(theDoUpToedState, 1, theEndState);
        aSATransition.setLabel(
            this.aDelayExpression.convertToKiel(
                anEsterelModule,
                theLocalEvents,
                theLocalVariables,
                null));
        aInitialArc.setSource(anInitialState);
        aInitialArc.setTarget(theDoUpToedState);
        // if theDoUpToedState is not DoUpToed
        //then a normal termination with lowest priority is used
        if (isFinalState) {
            return StateChartHelpers.createFinalOR(
                this.getEsterelStatementName() + "state",
                new Node[] {anInitialState, theEndState, theDoUpToedState });
        }
            return StateChartHelpers.createOR(
                this.getEsterelStatementName() + "state",
                new Node[] {anInitialState, theEndState, theDoUpToedState });

    }
    /**
     *Implemtents the abstrac methode from <code>EsterelStatement</code>
     * Parses an esterel statment in an <code>org.jdom.Document</code>
     * representation of an .exp file
     *  if you have not called the constructor
     * <code> EsterelStatement(EsterelParser theparser)</code>
     * use the methode <code>preParseEsterelStatement</code>
     * before <code>parseEsterelStatement</code>.
      *  <br>Sideeffects:
     * the class variables are set
     * Changes <code>EsterelParser</code>class variables
     * @param theparser an instance of <code>EsterelParser</code>
    * @throws EsterelParserException is only delivered
    * @see kiel.fileInterface.esterel.EsterelParserException
     * @see EsterelStatement#EsterelStatement(EsterelParser)
     *  @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see kiel.fileInterface.esterel.EsterelParser
     * @see org.jdom.Document
     */
    public final void parseEsterelStatement(final EsterelParser theparser)
        throws EsterelParserException {
        List anElementList =
```

```
        DOMHelpers.getElements(
            theparser.getXMLDocument(),
            this.getXPathSearchString());
    if (anElementList.size() == 2) {
        //the statement to DoUpTo
        theparser.incStatementCounter();
        // the delay expression
        this.aDelayExpression =
            new EsterelDelayExpression(
                theparser,
                "//*[@id='"
                    + ((Element) anElementList.get(1)).getAttributeValue(
                        "id")
                    + "']/*",
                theparser.getXMLDocument());
    } else {
        throw new EsterelParserException(
            this.getEsterelStatementName() + " number of elements wrong");
    }
} //public final void parseEsterelStatement(final EsterelParser theparser)
/**
* Fills the substatements in the esterel module.
* In this class it is fills the do Statements if there are any
*<br> Sideeffects: none
* @param theparser the actual parser
* @throws EsterelParserException is only delivered
* @see kiel.fileInterface.esterel.EsterelParserException
* @see kiel.fileInterface.esterel.EsterelParser
*/
public final void setSubStatments(final EsterelParser theparser)
    throws EsterelParserException {
    List anElementList =
        DOMHelpers.getElements(
            theparser.getXMLDocument(),
            this.getXPathSearchString());
    // the statement which may be DoUpTood is before the last element
    if (anElementList.size() == 2) {
        this.theStatement = theparser.getAddAt();
        theparser.getEsterelModule(
            theparser.getactualEsterModule())
            .addStatementToModule(
            theparser.getAddAt(),
            theparser.createStatement(((Element) (anElementList.get(0)))));
    }
}
/**
 *Implementents the abstrac methode from <code>EsterelStatement</code>.
* Returns a string representation of an esterel statement.
* @param anEsterelModule an esterel modul representation
* @return a String representation of an esterel statment
* @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
*@see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
*/
public final String toString(final EsterelModule anEsterelModule) {
    String result =
        "do \n"
            + (
                (EsterelStatement)
                    (
                        anEsterelModule.getEsterelProgram().get(
                            this.theStatement))).toString(
                    anEsterelModule);
    result += "\n upto " + this.aDelayExpression.toString(anEsterelModule);
    result += "\n end ";
    return result;
}
}
```

C.2.5. DoWatchingEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.5 dargestellt.

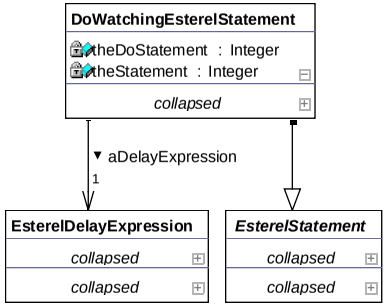


Abbildung C.5.: Klassediagramm `DoWatchingEsterelStatement`

Auflistung C.10: Die Klasse `DoWatchingEsterelStatement`

```
package kiel.fileInterface.esterel.esterel2estudio;
import java.util.ArrayList;
import java.util.List;
//
import kiel.dataStructure.FinalSimpleState;
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.Node;
import kiel.dataStructure.State;
import kiel.dataStructure.Transition;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StateChartHelpers;
//
import org.jdom.Element;
/**
 * <p>Implements the esterel <code>DoWatching</code> statement.</p>
 * <p>Copyright: Copyright (c) 2005</p>
 * <p>Company: Uni Kiel</p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
 * @version $Revision: 1.35 $ last modified $Date: 2006/02/06 18:35:28 $
 * <br>
 */
public class DoWatchingEsterelStatement extends EsterelStatement {
    /** The value of the Expression.*/
    private EsterelDelayExpression aDelayExpression;
    /**
     * the do statement.
     */
    private int theDoStatement;
    /**
     * the statment which to DoWatching.
     */
    private int theStatement;
    /**
     * the standart constructor.
     *
     */
    public DoWatchingEsterelStatement() {
        super();
    }
    /**
    * Parses a Document and set the class varaibles.
    * <br>It calls also the super constuctor<code>EsterelStatement
    * (EsterelParser theparser)</code> .
    * <br>Calls the <code>parseEsterelStatement</code> methode
    * <br>Sideeffects:
```

```
 * Changes <code>EsterelParser</code>class variables
*@param theparser an intace of a EsterelParser
* @throws EsterelParserException is only delivered
* @see kiel.fileInterface.esterel.EsterelParserException
* @see #parseEsterelStatement(EsterelParser)
* @see EsterelStatement#EsterelStatement(EsterelParser)
* @see kiel.fileInterface.esterel.EsterelParser
*/
public DoWatchingEsterelStatement(final EsterelParser theparser)
    throws EsterelParserException {
    super(theparser);
    this.theDoStatement = -1;
    this.theStatement = -1;
    if (theparser != null) {
        this.parseEsterelStatement(theparser);
    } else {
        throw new EsterelParserException("No Parser");
    }
}
/**
 * Returns a state representation of the esterel statement.
 * @param anEsterelModule the esterel module which
 * becomes a KIEL Statechart
 *@param isFinalState is true if the new state has to be
 * a final state
 * @param theLocalEvents Stack with the local events
 * @param theLocalVariables Stack with all local variables
* @param theTreesTrapSignals
 *                Stack with all tree trap signals.
 * @throws Esterel2EstudioException
 *                if the convertion to Kiel is not working. <br>
 *                Sideeffects: none
 * @return a State
 * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
 * @see kiel.dataStructure.State
 */
public final kiel.dataStructure.State convertToKiel(
       final EsterelModule anEsterelModule,
       final boolean isFinalState,
       final ArrayList theLocalEvents,
       final ArrayList theLocalVariables,
       final ArrayList theTreesTrapSignals)
       throws Esterel2EstudioException {
    InitialArc aInitialArc = new InitialArc();
    InitialState anInitialState = new InitialState();
    FinalSimpleState theNormalEnd = new FinalSimpleState();
    State theDoWatchingedState =
        (
            (EsterelStatement)
               (
                anEsterelModule.getEsterelProgram().get(
                   this.theStatement))).convertToKiel(
            anEsterelModule,
            false,
            theLocalEvents,
            theLocalVariables,
            theTreesTrapSignals);
    State theEndState;
    if (this.theDoStatement == -1) {
        theEndState = new FinalSimpleState();
    } else {
        theEndState =
            (
                (EsterelStatement)
                   (
                       anEsterelModule.getEsterelProgram().get(
                          this.theDoStatement))).convertToKiel(
                anEsterelModule,
                true,
                theLocalEvents,
                theLocalVariables,
                theTreesTrapSignals);
    }
    Transition aSATransition = null;
    aSATransition =
        StateChartHelpers.createSA(theDoWatchingedState, 1, theEndState);
    aSATransition.setLabel(
        this.aDelayExpression.convertToKiel(
            anEsterelModule,
            theLocalEvents,
            theLocalVariables,
            null));
    aInitialArc.setSource(anInitialState);
    aInitialArc.setTarget(theDoWatchingedState);
    // if theDoWatchingedState is not DoWatchinged
    //then a normal termination with lowest priority is used
    StateChartHelpers.createNT(theDoWatchingedState, 2, theNormalEnd);
    if (isFinalState) {
        return StateChartHelpers.createFinalOR(
            this.getEsterelStatementName() + "state",
            new Node[] {
                anInitialState,
                theEndState,
                theNormalEnd,
                theDoWatchingedState });
    }
        return StateChartHelpers.createOR(
            this.getEsterelStatementName() + "state",
            new Node[] {
                anInitialState,
                theEndState,
                theNormalEnd,
                theDoWatchingedState });

}
/**
 *Implemtents the abstrac methode from <code>EsterelStatement</code>
 * Parses an esterel statment in an <code>org.jdom.Document</code>
 * representation of an .exp file
 *  if you have not called the constructor
 * <code> EsterelStatement(EsterelParser theparser)</code>
 * use the methode <code>preParseEsterelStatement</code>
```

```
     * before <code>parseEsterelStatement</code>.
     *  <br>Sideeffects:
     * the class variables are set
     * Changes <code>EsterelParser</code>class variables
     * @param theparser an instance of <code>EsterelParser</code>
    * @throws EsterelParserException is only delivered
    * @see kiel.fileInterface.esterel.EsterelParserException
     * @see EsterelStatement#EsterelStatement(EsterelParser)
     *  @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see kiel.fileInterface.esterel.EsterelParser
     * @see org.jdom.Document
     */
    public final void parseEsterelStatement(final EsterelParser theparser)
        throws EsterelParserException {
        List anElementList =
            DOMHelpers.getElements(
                theparser.getXMLDocument(),
                this.getXPathSearchString());
        if (anElementList.size() > 1) {
            //the statement to DoWatching
            theparser.incStatementCounter();
            // the delay expression
            this.aDelayExpression =
                new EsterelDelayExpression(
                    theparser,
                    "//*[@id='"
                        + ((Element) anElementList.get(1)).getAttributeValue(
                            "id")
                        + "']/*",
                    theparser.getXMLDocument());
            // timeout statement
            if (anElementList.size() > 2) {
                theparser.incStatementCounter();
            }
        }
    } //public final void parseEsterelStatement(final EsterelParser theparser)
    /**
    * Fills the substatements in the esterel module.
    * In this class it is fills the do Statements if there are any
    *<br> Sideeffects: none
    * @param theparser the actual parser
    * @throws EsterelParserException is only delivered
    * @see kiel.fileInterface.esterel.EsterelParserException
    *  @see kiel.fileInterface.esterel.EsterelParser
    */
    public final void setSubStatments(final EsterelParser theparser)
        throws EsterelParserException {
        List anElementList =
            DOMHelpers.getElements(
                theparser.getXMLDocument(),
                this.getXPathSearchString());
        //  the statement which may be DoWatchinged is before the last element
        if (anElementList.size() > 1) {
            this.theStatement = theparser.getAddAt();
            theparser.getEsterelModule(
                theparser.getactualEsterModule())
                .addStatementToModule(
                theparser.getAddAt(),
                theparser.createStatement(((Element) (anElementList.get(0)))));
        }
        // the do statement
        if (anElementList.size() > 2) {
            this.theDoStatement = theparser.getAddAt();
            theparser.getEsterelModule(
                theparser.getactualEsterModule())
                .addStatementToModule(
                theparser.getAddAt(),
                theparser.createStatement(((Element) (anElementList.get(2)))));
        }
    }
    /**
     *Implemtents the abstrac methode from <code>EsterelStatement</code>.
    * Returns a string representation of an esterel statement.
    * @param anEsterelModule an esterel modul representation
    * @return a String representation of an esterel statment
    * @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
    *@see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
    */
    public final String toString(final EsterelModule anEsterelModule) {
        String result =
            "do \n"
                + (
                    (EsterelStatement)
                        (
                            anEsterelModule.getEsterelProgram().get(
                                this.theStatement))).toString(
                    anEsterelModule);
        result += "\n watching "
            + this.aDelayExpression.toString(anEsterelModule);
        if (this.theDoStatement != -1) {
            result += " timeout\n"
                + (
                    (EsterelStatement) anEsterelModule.getEsterelProgram().get(
                        this.theDoStatement)).toString(
                    anEsterelModule);
        }
        result += "\n end ";
        return result;
    }
}
```

C.2.6. EmitEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.6 dargestellt.

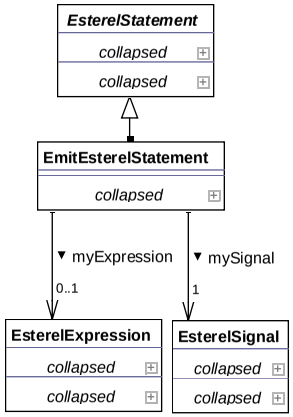


Abbildung C.6.: Klassediagramm `EmitEsterelStatement`

Auflistung C.11: Die Klasse `EmitEsterelStatement`

```
package kiel.fileInterface.esterel.esterel2estudio;

//
import java.util.ArrayList;
import java.util.List;

import kiel.dataStructure.CompoundLabel;
import kiel.dataStructure.FinalSimpleState;
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.Node;
import kiel.dataStructure.StringLabel;
import kiel.dataStructure.TransitionLabel;
import kiel.dataStructure.action.Actions;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StateChartHelpers;
import org.jdom.Element;

/**
 * <p>
 *  This is a subclass of <code>EsterelStatement</code>
 * classes implements the emit statement.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2004
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.58 $ last modified $Date: 2006/02/13 14:00:04 $
 */
public class EmitEsterelStatement
       extends EsterelStatement {
   /** The expression of the signal to emit. */
```

```
40      private EsterelExpression myExpression;

        /** The signal to emit. */
        private EsterelSignal mySignal = null;

        /**
         * Simple constructor.
         */
        public EmitEsterelStatement() {
            super();
50          this.myExpression = null;
            this.mySignal = null;
        } //   public EmitEsterelStatement()

        /**
         * Parses a Document and set the class varaibles. <br>
         * It calls also the super constuctor <code>EsterelStatement
         * (EsterelParser theparser)</code>.
         * <br>
         * Calls the <code>parseEsterelStatement</code> methode <br>
60       * Sideeffects: Changes <code>EsterelParser</code> class variables
         *
         * @param theparser
         *                  an intace of a EsterelParser
         * @throws EsterelParserException
         *                  is only delivered
         * @see kiel.fileInterface.esterel.EsterelParserException
         * @see EsterelStatement#parseEsterelStatement(EsterelParser)
         * @see EsterelStatement#EsterelStatement(EsterelParser)
         * @see kiel.fileInterface.esterel.EsterelParser
70       */
        public EmitEsterelStatement(final EsterelParser theparser)
                throws EsterelParserException {
            super(
                    theparser);
            this.parseEsterelStatement(theparser);
        } //public EmitEsterelStatement(final EsterelParser theparser)

        /**
         * Returns a state representation of the esterel statement.
80       *
         * @param anEsterelModule
         *                  the esterel module which becomes a KIEL Statechart
         * @param isFinalState
         *                  is true if the new state has to be a final state
         * @param theLocalEvents
         *                  Stack with the local events
         * @param theLocalVariables
         *                  Stack with all local variables
         * @param theTreesTrapSignals
90       *                  Stack with all tree trap signals.
         * @throws Esterel2EstudioException
         *                  if the convertion to Kiel is not working. <br>
         *                  Sideeffects: none
         * @return a State
         * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
         * @see kiel.dataStructure.State
         */
        public final kiel.dataStructure.State convertToKiel(
                final EsterelModule anEsterelModule,
100             final boolean isFinalState,
                final ArrayList theLocalEvents,
                final ArrayList theLocalVariables,
                final ArrayList theTreesTrapSignals)
                throws Esterel2EstudioException {
            InitialArc aInitialArc = new InitialArc();
            InitialState anInitialState = new InitialState();
            FinalSimpleState theEndState = new FinalSimpleState();
            aInitialArc.setSource(anInitialState);
            aInitialArc.setTarget(theEndState);
110         TransitionLabel theLabelofaTransition = null;
            String aLabel = this.mySignal.getSignalIdentifier();
            if (this.myExpression != null) {
            aLabel +=
             "( "
            + this.myExpression.toString(anEsterelModule)
            + ")";
            }
            Actions theActions = StateChartHelpers
                    .getActions(anEsterelModule.getEsterelStateChart(),
120                         theLocalVariables,
                            theLocalEvents,
                            " " + aLabel);
                if (theActions == null) {
                   theLabelofaTransition = new StringLabel("/" + aLabel);
                }
                if (theLabelofaTransition == null) {
                theLabelofaTransition = new CompoundLabel();
                ((CompoundLabel) theLabelofaTransition).setEffect(theActions);
                }
130         aInitialArc.setLabel(theLabelofaTransition);
            if (isFinalState) {
                return StateChartHelpers
                .createFinalOR(this.getEsterelStatementName() + "state",
                    new Node[] {
                            anInitialState, theEndState });
            }
            return StateChartHelpers
            .createOR(this.getEsterelStatementName() + "state",
                new Node[] {
140                 anInitialState, theEndState });
        } //public final kiel.dataStructure.State convertToKiel(

        /**
         * Implemtents the abstrac methode from <code>EsterelStatement</code>.
         * Parses an esterel statment in an <code>org.jdom.Document</code>
         * representation of an .exp file. if you have not called the
         * constructor <code> EsterelStatement(EsterelParser theparser)</code>
         * use the methode <code>preParseEsterelStatement</code> before
         * <code>parseEsterelStatement</code><br>
150      * Sideeffects: the class variables are set Changes
         * <code>EsterelParser</code> class variables
```

```
     *
     * @param theparser
     *                an instance of <code>EsterelParser</code>
     * @throws EsterelParserException
     *                is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     * @see EsterelStatement#EsterelStatement(EsterelParser)
     * @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see EsterelStatement#preParseEsterelStatement(EsterelParser)
     * @see kiel.fileInterface.esterel.EsterelParser
     * @see org.jdom.Document
     */
    public final void parseEsterelStatement(
           final EsterelParser theparser)
           throws EsterelParserException {
       final int toomuch = 3;
       List anElementList = null;
       anElementList =
           DOMHelpers.getElements(theparser.getXMLDocument(),
           this.getXPathSearchString());
       if (anElementList.size() > 0 && anElementList.size() < toomuch) {
           this.mySignal =
               theparser
               .getEsterelModule(theparser
                   .getactualEsterModule())
               .getEsterelSignalByID(
                   this.getSignal(anElementList));
           if (anElementList.size() > 1 && anElementList.size() < toomuch) {
               final int theIndex = 1;
               this.myExpression = new EsterelExpression(
                   theparser,
                       this.getXPathSearchString() + "[local-name(.)='"
                              + ((Element) anElementList.get(theIndex))
                              .getName()
                              + "'][@id='"
                              + ((Element) anElementList.get(theIndex))
                              .getAttributeValue("id")
                              + "']/*",
                       theparser.getXMLDocument());
           } //if >1
       } else {
           throw new EsterelParserException(
                   this.getEsterelStatementName() + " no signal expression",
                   theparser);
       } //if >0
    } //public final void parseEsterelStatement(final EsterelParser

    // theparser) {
    /**
     * Fills the substatements in the esterel module in this class it is
     * empty. <br>
     * Sideeffects: none
     *
     * @param theparser
     *                the actual parser
     * @throws EsterelParserException
     *                is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     * @see kiel.fileInterface.esterel.EsterelParser
     */
    public final void setSubStatments(
           final EsterelParser theparser)
           throws EsterelParserException {
    }

    //final EsterelModule anEsterelModule)
    /**
     * Implemtents the abstrac methode from <code>EsterelStatement</code>
     * Returns a string representation of an esterel statement.
     *
     * @param anEsterelModule
     *                an esterel modul representation
     * @return a String representation of an esterel statment
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    public final String toString(
           final EsterelModule anEsterelModule) {
       return this.getEsterelStatementName() + " "
              + this.mySignal
                     .getSignalIdentifier()
              + " "
              + this.myExpression;
    }

    /**
     * Retuns the signal of the Signalexpession. <br>
     * Sideeffects: none
     *
     * @param anElementList
     *                the cildren elements of the SignalExpression
     * @return the signals id
     * @throws EsterelParserException
     *                if no signal is found
     */
    private String getSignal(
           final List anElementList)
           throws EsterelParserException {
       final int theSignalIndex = 0;
       if (anElementList.size() > theSignalIndex) {
           return ((Element) anElementList
                  .get(theSignalIndex))
           .getAttributeValue("id");
       }
       throw new EsterelParserException(
              this.getEsterelStatementName() + ": found no signal ");
    }
} // end of class
```

C.2.7. Esterel2EstudioException

Auflistung C.12: Die Klasse `Esterel2EstudioException`

```
package kiel.fileInterface.esterel.esterel2estudio;
import javax.swing.JOptionPane;

/**
 * <p>
 *  The Exceptionclass for the EsterelParser.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 */
public class Esterel2EstudioException extends Exception {
    /**
     * the standart constructor.
     */
    public Esterel2EstudioException() {
        super();
    }

    /**
     * @param arg0
     *                the message
     */
    public Esterel2EstudioException(final String arg0) {
        super(arg0);
    }

    /**
     * Shows a error pane.
     */
    public final void showMessageBox() {
        JOptionPane.showMessageDialog(
                null,
                this.getLocalizedMessage(),
                "Error",
                JOptionPane.ERROR_MESSAGE);
    }
}
```

C.2.8. Esterel2EstudioProperties

Auflistung C.13: Die Klasse `Esterel2EstudioProperties`

```
/*
 * $Author: lku $ $Date: 2006/02/06 18:35:29 $
 */
package kiel.fileInterface.esterel.esterel2estudio;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;

/**
 * <p>
 *  Used to load and store user defineable properties.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars K&uumlhl </a>
 * @version $Revision: 1.11 $
 */
public final class Esterel2EstudioProperties {
    /**
     * The default ressource for the properties.
     */
    private static final String DEFAULTRESOURCE = "esterel2estudio.properties";
    /**
     * The name in the properties file..
     */
    private static final String FUNCTIONS = "Functions";
    /**
     * The name in the properties file..
     */
    private static final String SEPERATOR = "ArgumentSeperator";
    /**
     * The name in the properties file.
     */
    private static final String PROCEDURES = "Procedures";
    /**
     * The name in the properties file..
     */
    private static final String TASKS = "Tasks";
    /**
     * @return true, if option is set to true.
     */
    public static boolean getFunctions() {
        return properties.getProperty(FUNCTIONS)
                .equalsIgnoreCase("true");
    }
    /**
     * @return true, if option is set to true.
     */
    public static boolean getProcedures() {
        return properties.getProperty(PROCEDURES)
                .equalsIgnoreCase("true");
    }
    /**
     * @return true, if option is set to true.
     */
    public static boolean getTasks() {
        return properties.getProperty(TASKS)
                .equalsIgnoreCase("true");
    }

    /**
     * The propertie key for the pre string for new signals or variables
     * which where created in the convertion.
     */
    private static final String NEWSIGNALSPOSTSTRING = "PostString";

    /**
     * The propertie key for the pre string for new signals or variables
     * which where created in the convertion..
     */
    private static final String NEWSIGNALSPRESTRING = "PreString";

    /**
     * These are the internal properties.
     */
    private static Properties properties;

    /**
     * This is the key for user specific file.
     */
    private static final String PROPERTIESFILE = System.getProperty("user.home")
            + File.separator
            + ".kiel"
            + File.separator
            + "esterel2estudio.properties";

    /**
     * @return the post string for new signals
    */
    public static String getPostString() {
        return properties.getProperty(NEWSIGNALSPOSTSTRING);
    }
    /**
     * @return the seperator in calls
```

```
     */
    public static String getSeperatorString() {
        return properties.getProperty(SEPERATOR);
    }
    /**
     * @return if automatic optimization is on means true.
     */
    public static String getPreSting() {
        return properties.getProperty(NEWSIGNALSPRESTRING);
    }

    /**
     * This method will load the properties from local file. If this file
     * does not exist, the defaults are written to local file.
     *
     * @return true, if loading was successful
     */
    public static boolean load() {
        boolean success;
        properties = new Properties(
                getDefaults());
        if (!loadProperties()) {
            success = copyDefaults();
        } else {
            success = true;
        }
        return success;
    }

    /**
     * Reloads the properties file.
     */
    public static void reload() {
        loadProperties();
    }

    /**
     * Copies properties to user specific file.
     *
     * @return true, if copy was successful
     */
    private static boolean copyDefaults() {
        InputStream is;
        FileWriter fw;
        boolean success = true;
        try {
            is = Esterel2EstudioProperties
            .class
            .getResourceAsStream(DEFAULTRESOURCE);
            fw = new FileWriter(
                    PROPERTIESFILE);
            int c = is.read();
            while (c >= 0) {
                fw.write(c);
                c = is.read();
            }
            fw.flush();
            is.close();
            fw.close();
        } catch (IOException storeExeception) {
            success = false;
        }
        return success;
    }

    /**
     * Loads the defaults form resource file.
     *
     * @return the default values
     */
    private static Properties getDefaults() {
        Properties defaults = new Properties();
        try {
            defaults
            .load(Esterel2EstudioProperties
                    .class
                    .getResourceAsStream(DEFAULTRESOURCE));
        } catch (IOException e) {
            defaults.clear();
        }
        return defaults;
    }

    /**
     * Loads properties from user specific file.
     *
     * @return true, if loading was successful
     */
    private static boolean loadProperties() {
        boolean success = true;
        try {
            properties.load(new FileInputStream(
                    PROPERTIESFILE));
        } catch (IOException loadException) {
            success = false;
        }
        return success;
    }

    /**
     * This is the construtor.
     */
    private Esterel2EstudioProperties() {
    }
}
```

C.2.9. EsterelConstant

Der Klassenaufbau ist in der Abbildung C.2.9 dargestellt.

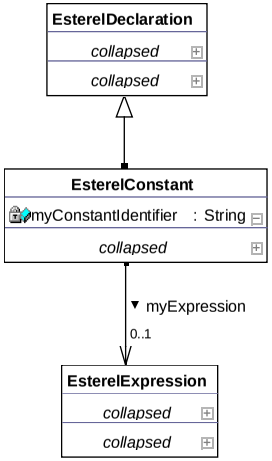


Abbildung C.7.: Klassediagramm `EsterelConstant`

Auflistung C.14: Die Klasse `EsterelConstant`

```
package kiel.fileInterface.esterel.esterel2estudio;
//
import java.util.ArrayList;
import java.util.List;
import kiel.dataStructure.StringVariable;
import kiel.dataStructure.Variable;
import kiel.dataStructure.boolexp.BooleanVariable;
import kiel.dataStructure.doubleexp.DoubleVariable;
import kiel.dataStructure.floatexp.FloatVariable;
import kiel.dataStructure.intexp.IntegerVariable;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import org.jdom.Element;
/**
 * <p> It represents esterel constants .</p>
 *  <p>Copyright: Copyright (c) 2005</p>
 * <p>Company: Uni Kiel</p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
 * @version $Revision: 1.35 $ last modified $Date: 2006/02/06 18:35:29 $
 * <br>
 */
public class EsterelConstant
extends EsterelDeclaration {
    /**
    * The type of the constant.(Optional).
    */
    private EsterelTypeDeclaration myChannelType; // child1
    /**
     *The name of the constant.
     */
```

```
private String myConstantIdentifier; // child 0
/**
 *The expression of the constant.(Optional).
 */
private EsterelExpression myExpression; //child 2
/**
 * the standart constructor.
 */
public EsterelConstant() {
    super();
}
/**
 * Creates an EsterelVariable and fills the class variables.
 * <br>Sideeffects: class variables are changed.
 * @param theparser the esterel parser
 * @param theXPath the XPath to the EsterelVariable
 * @throws EsterelParserException if something
 * goes wrong at the parsing.
 */
public EsterelConstant(
    final EsterelParser theparser,
    final String theXPath)
    throws EsterelParserException {
    super();
    this.myConstantIdentifier = null;
    this.myExpression = null;
    this.myChannelType = null;
    if (theparser != null) {
        parseEsterelStatement(theparser, theXPath);
    }
}
/**
 * Returns a constant representation of the esterel constant.
 *
 * @param anEsterelModule
 *                  the esterel module which becomes a KIEL Statechart
 * @param theLocalEvents
 *                  Stack with the local events
 * @param theLocalVariables
 *                  Stack with all local variables
 * @throws Esterel2EstudioException
 *                  if the convertion to Kiel is not working. <br>
 *                  Sideeffects: none
 * @return a State
 * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
 * @see kiel.dataStructure.State
 */
public final Variable convertToKiel(
       final EsterelModule anEsterelModule,
       final ArrayList theLocalEvents,
       final ArrayList theLocalVariables)
       throws Esterel2EstudioException {
   Variable result = null;
   if (result == null) {
       if (this.myChannelType != null) {
           if (this.myChannelType
                   .getTypeName()
                   .compareTo("string") == 0) {
               if (this.myExpression != null) {
                   result = new StringVariable(
                           this.myConstantIdentifier,
                           this.myExpression
                           .convertToKielString(anEsterelModule));
               } else {
                   result = new StringVariable(
                           this.myConstantIdentifier);
               }
           } else if (this.myChannelType
                   .getTypeName()
                   .compareTo("integer") == 0) {
               if (this.myExpression != null) {
                   result = new IntegerVariable(
                           this.myConstantIdentifier,
                           this.myExpression.convertToKiel(anEsterelModule,
                               theLocalEvents,
                               theLocalVariables));
               } else {
                   result = new IntegerVariable(
                           this.myConstantIdentifier);
               }
           } else if (this.myChannelType
                   .getTypeName()
                   .compareTo("boolean") == 0) {
               if (this.myExpression != null) {
                   result = new BooleanVariable(
                           this.myConstantIdentifier,
                           this.myExpression
                           .convertToKielBoolExp(anEsterelModule,
                               theLocalEvents,
                               theLocalVariables));
               } else {
                   result = new BooleanVariable(
                           this.myConstantIdentifier);
               }
           } else if (this.myChannelType
                   .getTypeName()
                   .compareTo("float") == 0) {
               if (this.myExpression != null) {
                   result = new FloatVariable(
                           this.myConstantIdentifier);
                   /*
                    * After full suport for Float this.myExpression
                    * .convertToKiel(anEsterelModule));
                    */
               } else {
                   result = new FloatVariable(
                           this.myConstantIdentifier);
               }
           } else if (this.myChannelType
                   .getTypeName()
                   .compareTo("double") == 0) {
               if (this.myExpression != null) {
```

```
                    result = new DoubleVariable(
                            this.myConstantIdentifier);
                    /*
                     * After full suport for double:
                     * /*this.myExpression
                     * .convertToKiel(anEsterelModule));
                     */
                } else {
                    result = new DoubleVariable(
                            this.myConstantIdentifier);
                }
            } else {
                // set the type to string
                if (this.myExpression != null) {
                    result = new StringVariable(
                            this.myConstantIdentifier,
                            this.myExpression
                            .convertToKielString(anEsterelModule));
                    /*
                     * throw new Esterel2EstudioException(
                     * this.getName() + "Unknown type : " +
                     * this.myChannelType);
                     */
                } else {
                    result = new StringVariable(
                            this.myConstantIdentifier, "");
                }
            }
        } else {
            throw new Esterel2EstudioException(
                    this.getName() + ": no Type");
        }
    }
    return result;
}

/**
 * Parses a variable. <br>
 * Sideeffects: sets the class variables.
 *
 * @param theXPath
 *                the XPath which leads to the Expression
 * @param theparser
 *                the esterel parser
 * @throws EsterelParserException
 *                if there is a parsing problem
 * @see kiel.fileInterface.esterel.EsterelParserException
 */
public final void parseEsterelStatement(
        final EsterelParser theparser,
        final String theXPath)
        throws EsterelParserException {
    List anElementList = DOMHelpers.getElements(theXPath,
        theparser.getXMLDocument());
    String theName = ((Element) anElementList.get(0))
    .getParentElement()
    .getName();
    if (anElementList.size() > 0
            && theName.compareTo("ConstantSymbol") != 0
                    ) {
        throw new EsterelParserException(
                "EsterelConstant"
                        + " Unknown Type : "
                        + ((Element) anElementList.get(0))
                        .getParentElement()
                                .getName());
    }
    this.setName(((Element) anElementList.get(0)).getParentElement()
    .getName());
    this.setXMLID(((Element) anElementList.get(0)).getParentElement()
        .getAttributeValue("id"));
    for (int i = 0; i < anElementList.size(); i++) {
        if (i == 0
                && ((Element) anElementList.get(i)).getName() != "NULL") {
            this.myConstantIdentifier = new String(
                    ((Element) anElementList.get(i)).getText());
        } else if (i == 1
                && ((Element) anElementList.get(i)).getName() != "NULL") {
            this.myChannelType =
                theparser
                .gettheEsterelModules()[theparser.getactualEsterModule()]
                .getEsterelTypeByID(((Element) anElementList.get(i))
                .getAttributeValue("id"));
        } else if (i == 2
                && ((Element) anElementList.get(i)).getName() != "NULL") {
            this.myExpression = new EsterelExpression(
                theparser,
                    theXPath
                            + "[local-name(.)='"
                            + ((Element) anElementList.get(i)).getName()
                            + "'][@id='"
                            + ((Element) anElementList.get(i))
                            .getAttributeValue("id")
                            + "']/*", theparser.getXMLDocument());
        } else if (i == 0) {
            throw new EsterelParserException(
                    "No Constantname");
        }
    }
}
/**
 * Returns the String representation of the variable.
 * @param anEsterelModule an esterel module.
 * @return the String representation of an EsterelVariable
 */
public final String toString(final EsterelModule anEsterelModule) {
    String result = this.myConstantIdentifier;
    if (this.myExpression != null) {
        result += " := " + this.myExpression.toString(anEsterelModule);
    }
    if (this.myChannelType != null) {
        result += " : " + this.myChannelType;
```

```
    }
    return result + "\n";
}

/**
 * @return Returns the myChannelType.
 */
public final String getChannelType() {
    return this.myChannelType.getTypeName();
}
/**
 * @return Returns the myConstantIdentifier.
 */
public final String getConstantIdentifier() {
    return this.myConstantIdentifier;
}
/**
 * @return Returns the myExpression.
 */
public final EsterelExpression getExpression() {
    return this.myExpression;
}
}
```

C.2.10. EsterelDeclaration

Der Klassenaufbau ist in der Abbildung C.2.10 dargestellt.

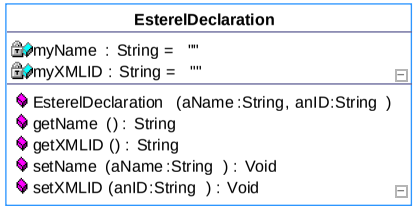


Abbildung C.8.: Klassediagramm `EsterelDeclaration`

Auflistung C.15: Die Klasse `EsterelDeclaration`

```
package kiel.fileInterface.esterel.esterel2estudio;

/**
 * <p>
 *  The superclass for all Esterel classes.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.5 $ last modified $Date: 2006/02/06 18:35:29 $
 *               <br>
 */
public class EsterelDeclaration {
/**
 *  the name.
 */
   private String myName = "";

   /**
    * the xml id in a file.
    */
   private String myXMLID = "";

/**
 * the standard constructor.
 *
 */
   public EsterelDeclaration() {
       this(
               "", "");
   }
/**
 *  sets the class variables in the constructor.
 * @param aName a name
 * @param anID an id
 */
   public EsterelDeclaration(final String aName,
           final String anID) {
       super();
       this.myName = aName;
       this.myXMLID = anID;
   }

   /**
    * @return Returns the myName.
    */
   public final String getName() {
       return this.myName;
   }

   /**
    * @return Returns the myID.
    */
```

```
    public final String getXMLID() {
      return this.myXMLID;
    }

    /**
     * @param aName          The aName to set.
     *
     */
    public final void setName(
        final String aName) {
      this.myName = aName;
    }

    }

    /**
     * @param anID           The anID to set.
     *
     */
    public final void setXMLID(
        final    String anID) {
      this.myXMLID = anID;
    }
  }
```

C.2.11. EsterelDeclarationSortByID

Auflistung C.16: Die Klasse `EsterelDeclarationSortByID`

```
package kiel.fileInterface.esterel.esterel2estudio;

import java.util.Comparator;

/**
 * <p>
 *  Defines an order to EsterelDeclarations by XMLID.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.13 $ last modified $Date: 2006/02/07 15:39:35 $
 *              <br>
 */
public final class EsterelDeclarationSortByID
       implements Comparator {
   /**
    * The standart constructor.
    */
   public EsterelDeclarationSortByID() {
       super();
   }

   /**
    * @param arg0
    *                 an EsterelDeclaration
    * @param arg1
    *                 an EsterelDelcaration
    * @return <p>0 if arg0==arg1, neagtive value if arg0 < arg1, positive
    *            value if arg1 < arg0.</p>
    * @see kiel.fileInterface.esterel.esterel2estudio.EsterelDeclaration
    */
   public int compare(
           final EsterelDeclaration arg0,
           final EsterelDeclaration arg1) {
       int result = 0;
           if (arg0.getXMLID().length() < arg1.getXMLID().length()) {
               result = -1;
           } else if (arg0.getXMLID().length() > arg1.getXMLID().length()) {
               result = 1;
           } else {
               result = arg0.getXMLID().compareTo(arg1.getXMLID());
           }
       return result;
   }
   /**
    * Works only with EsterelDeclarations. If arg0 or arg1
    * are not instanceof EsterelDelaration 0 is returned.
    * @param arg0
    *                 an EsterelDeclaration
    * @param arg1
    *                 an EsterelDelcaration
    * @return <p>0 if arg0==arg1, neagtive value if arg0 < arg1, positive
    *            value if arg1 < arg0.</p>
    * @see kiel.fileInterface.esterel.esterel2estudio.EsterelDeclaration
    */
   public int compare(
     final     Object arg0,
     final     Object arg1) {
       if (arg0 instanceof EsterelDeclaration
               && arg1 instanceof EsterelDeclaration) {
           return this.compare((EsterelDeclaration) arg0,
               (EsterelDeclaration) arg1);
       }
       return 0;
   }
}
```

C.2.12. EsterelDeclarationSortByName

Auflistung C.17: Die Klasse `EsterelDeclarationSortByName`

```
package kiel.fileInterface.esterel.esterel2estudio;

import java.util.Comparator;

/**
 * <p>
 *  Defines an order to EsterelDeclarations by name.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.12 $ last modified $Date: 2006/02/06 18:35:29 $
 *              <br>
 */
public final class EsterelDeclarationSortByName
       implements Comparator {
   /**
    * The standart constructor.
    */
   public EsterelDeclarationSortByName() {
      super();
   }

   /**
    * @param arg0
    *                an EsterelDeclaration
    * @param arg1
    *                an EsterelDelcaration
    * @return <p>0 if arg0==arg1, neagtive value if arg0 < arg1, positive
    *            value if arg1 < arg0.</p>
    * @see kiel.fileInterface.esterel.esterel2estudio.EsterelDeclaration
    */
   public int compare(
          final EsterelDeclaration arg0,
          final EsterelDeclaration arg1) {
      return arg0.getName().compareTo(arg1.getName());
   }


   /**
    * Works only with EsterelDeclarations. If arg0 or arg1
    * are not instanceof EsterelDelaration 0 is returned.
    * @param arg0
    *                an EsterelDeclaration
    * @param arg1
    *                an EsterelDelcaration
    * @return <p>0 if arg0==arg1, neagtive value if arg0 < arg1, positive
    *            value if arg1 < arg0.</p>
    * @see kiel.fileInterface.esterel.esterel2estudio.EsterelDeclaration
    */
   public int compare(
     final     Object arg0,
     final     Object arg1) {
       if (arg0 instanceof EsterelDeclaration
              && arg1 instanceof EsterelDeclaration) {
          return this.compare((EsterelDeclaration) arg0,
              (EsterelDeclaration) arg1);
       }
       return 0;
   }
}
```

C.2.13. EsterelDelayExpression

Der Klassenaufbau ist in der Abbildung C.2.13 dargestellt.

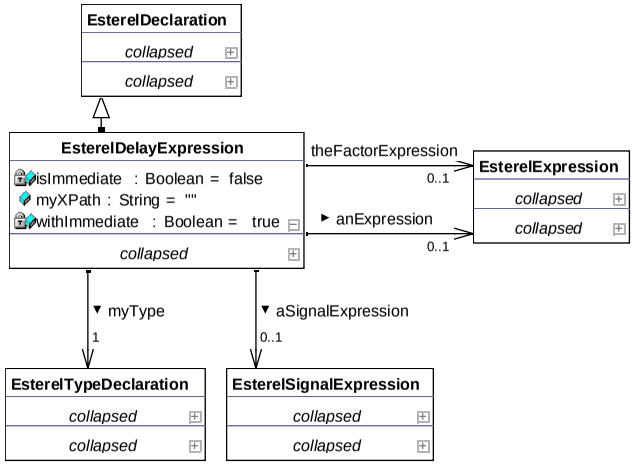


Abbildung C.9.: Klassediagramm `EsterelDelayExpression`

Auflistung C.18: Die Klasse `EsterelDelayExpression`

```
package kiel.fileInterface.esterel.esterel2estudio;

import java.util.ArrayList;
import java.util.List;

import kiel.dataStructure.CompoundLabel;
import kiel.dataStructure.StringLabel;
import kiel.dataStructure.TransitionLabel;
import kiel.dataStructure.eventexp.DelayExpression;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.CompoundLabelException;
import kiel.util.CompoundLabelParser;
import kiel.util.DOMHelpers;
import org.jdom.Document;
import org.jdom.Element;

/**
 * <p>
 *  Represents a SignalExpression in esterel.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
```

```
 * @version $Revision: 1.33 $ last modified $Date: 2006/02/07 15:39:35 $
 */
public class EsterelDelayExpression extends EsterelDeclaration {
    /**
     * the Expression part of the DelayExpression.
     */
    private EsterelExpression anExpression;

    /**
     * the SignalExpression part of the DelayExpression.
     */
    private EsterelSignalExpression aSignalExpression;

    /**
     * is it immediate.
     */
    private boolean isImmediate;

    /**
     * the type of the DelayExpression.
     */
    private EsterelTypeDeclaration myType;

    /**
     * the XPath to the DelayExpression.
     */
    private String myXPath;

    /**
     * The factor expression.
     */
    private EsterelExpression theFactorExpression;
    /** Only for coding reasons no exteren access.*/
    private boolean withImmediate = true;

    /**
     * the standart constructor.
     */
    public EsterelDelayExpression() {
        super();
        this.theFactorExpression = null;
        this.anExpression = null;
        this.isImmediate = false;
        this.aSignalExpression = null;
        this.myXPath = "";
        this.myType = null;
    }

    /**
     * Uses the standard constructor and call the parseDelayExpression
     * method. <br>
     * Sideeffects: sets the class variables
     *@param theparser an esterel parser
     * @param aXPath
     *                the XPath which leads to this Signalexpression in the
     *                <br>
     *                <code>org.jdom.Document</code> respresentation of an
     *                .exp file
     * @param xmldoc
     *                the <code>org.jdom.Document</code>
     * @throws EsterelParserException
     *                 is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     */
    public EsterelDelayExpression(final EsterelParser theparser,
            final String aXPath,
            final Document xmldoc)
            throws EsterelParserException {
        this();
        this.parseDelayExpression(theparser, aXPath, xmldoc);
    }

    /**
     * Converts the delayexpression to a
     * <code>kiel.dataStructure.TransitionLabel</code>.
     *
     * @param anEsterelModule
     *                the esterel module
     * @param theLocalEvents
     *                all local events
     * @param theLocalVariables
     *                all local variables
     * @param aTransLabel
     *                an old label or null
     * @return a <code>kiel.dataStructure.TransitionLabel</code>
     * @see kiel.dataStructure.TransitionLabel
     */
    public final TransitionLabel convertToKiel(
            final EsterelModule anEsterelModule,
                    final ArrayList theLocalEvents,
                    final ArrayList theLocalVariables,
            final TransitionLabel aTransLabel) {
        this.withImmediate = false;
        CompoundLabelParser.getInstance();
        // Setting StateChart for identifier searching
        CompoundLabelParser.setStateChart(anEsterelModule
                .getEsterelStateChart());
        CompoundLabelParser.setAllLocals(theLocalVariables, theLocalEvents);
        String aStringLabel = this.toString(anEsterelModule).trim();
        this.withImmediate = true;
        TransitionLabel aLabel = null;
        DelayExpression aDelayExp = null;
        try {
            aDelayExp = CompoundLabelParser.parseDelayExpression(
                    aStringLabel,
                    this.isImmediate);
        } catch (CompoundLabelException ex) {
            return aLabel = new StringLabel(aStringLabel);
        }
        if (aTransLabel != null) {
            if (aTransLabel instanceof StringLabel) {
                aLabel = new StringLabel(aStringLabel);
```

```
            } else {
                aLabel = aTransLabel;
                ((CompoundLabel) aLabel).setTrigger(aDelayExp);
            }
        } else {
            aLabel = new CompoundLabel();
            ((CompoundLabel) aLabel).setTrigger(aDelayExp);
        }
        return aLabel;
    } // covert
    /**
     * @return Returns the anExpression.
     */
    public final EsterelExpression getAnExpression() {
        return this.anExpression;
    }
    /**
     * @return Returns the aSignalExpression.
     */
    public final EsterelSignalExpression getASignalExpression() {
        return this.aSignalExpression;
    }
    /**
     * @return Returns the myType.
     */
    public final String getMyType() {
        return this.myType.getTypeName();
    }
    /**
     * @return Returns the myXPath.
     */
    public final String getMyXPath() {
        return this.myXPath;
    }
    /**
     * @return Returns the theFactorExpression.
     */
    public final EsterelExpression getTheFactorExpression() {
        return this.theFactorExpression;
    }
    /**
     * @return Returns the isImmediate.
     */
    public final boolean isImmediate() {
        return this.isImmediate;
    }

    /**
     * Parses a DelayExpression. <br>
     * Sideeffects: sets the class variables
     *
     *@param theparser an <code>EsterelParser</code>
     * @param aXPath
     *                 the XPath which leads to the DelayExpression
     * @param xmldoc
     *                 the <code>org.jdom.Document</code> reprsentation of the
     *                 .exp file
     * @throws EsterelParserException
     *                 is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     */
    public final void parseDelayExpression(
            final EsterelParser theparser,
            final String aXPath,
            final Document xmldoc) throws EsterelParserException {
        int i = 0;
        this.myXPath = aXPath;
        List aDelayElementList = DOMHelpers.getElements(this.myXPath
                + "[not(local-name(.)='EOV')]", xmldoc);
        // the XPath finds the children of the SignalExpression
        String actualElement = ((Element) aDelayElementList.get(i))
                .getParentElement().getName();
        this.setXMLID(((Element) aDelayElementList.get(i))
                .getParentElement().getAttributeValue("id"));
        this.setName(actualElement);
        if (actualElement == "Delay") {
            int j = 0; // 0tes Element Type of Delay
            if (((Element) aDelayElementList.get(j)).getName() != "NULL") {
                this.myType =
                    theparser
                    .gettheEsterelModules()[theparser.getactualEsterModule()]
                    .getEsterelTypeByID(((Element) aDelayElementList.get(j))
                    .getAttributeValue("id"));
                j++;
            } else {
                j++;
            }
            // 1tesElement SignalExpression
            if (((Element) aDelayElementList.get(j)).getName() != "NULL") {
                this.aSignalExpression = new EsterelSignalExpression(
                    theparser,
                        this.myXPath
                                + "[local-name(.)='"
                                + ((Element) aDelayElementList.get(j))
                                        .getName()
                                + "'][@id='"
                                + ((Element) aDelayElementList.get(j))
                                        .getAttributeValue("id") + "']/*",
                        xmldoc);
                j++;
            } else {
                j++;
            }
            //2tesElement Expression
            if (((Element) aDelayElementList.get(j)).getName() != "NULL") {
                this.anExpression = new EsterelExpression(
                    theparser,
                    this.myXPath
                        + "[local-name(.)='"
                        + ((Element) aDelayElementList.get(j)).getName()
                        + "'][@id='"
                        + ((Element) aDelayElementList.get(j))
```

```
                        .getAttributeValue("id") + "']/*", xmldoc);
                j++;
            } else {
                j++;
            }
            /* 3tesElement immediate */
            if (((Element) aDelayElementList.get(j)).getName() != "NULL") {
                if (((Element) aDelayElementList.get(j)).getName()
                        == "BoolTrue") {
                    this.isImmediate = true;
                } else {
                    this.isImmediate = false;
                }
                j++;
            } else {
                this.isImmediate = false;
                j++;
            }
            /*
             * It is a reference to a couter. We dont need ist.
             */
            if (((Element) aDelayElementList.get(j)).getName() != "NULL") {
                j++;
            } else {
                j++;
            }
        } else if (actualElement == "LoadSignalExpression"
                || actualElement == "BinaryOp"
                || actualElement == "UnaryOp") {
            this.aSignalExpression = new EsterelSignalExpression(
                theparser,
                this.myXPath,
                    xmldoc);
        } else {
            throw new EsterelParserException(
                    "DelayExpression: cannot identify expression : "
                            + actualElement + this.getXMLID());
        }
    }
    /**
     * @param anExp The anExpression to set.
     */
    public final void setAnExpression(final EsterelExpression anExp) {
        this.anExpression = anExp;
    }
    /**
     * @param signalExpression The aSignalExpression to set.
     */
    public final void setASignalExpression(
            final EsterelSignalExpression signalExpression) {
        this.aSignalExpression = signalExpression;
    }
    /**
     * @param anisImmediate The isImmediate to set.
     */
    public final void setImmediate(final boolean anisImmediate) {
        this.isImmediate = anisImmediate;
    }

    /**
     * @param aXPath The myXPath to set.
     */
    public final void setMyXPath(final String aXPath) {
        this.myXPath = aXPath;
    }
    /**
     * @param aFactorExpression The theFactorExpression to set.
     */
    public final void setTheFactorExpression(
            final EsterelExpression aFactorExpression) {
        this.theFactorExpression = aFactorExpression;
    }

    /**
     * Creats the String representation.
     *@param anEsterelModule an <code>EsterelModule</code>
     * @return the String representation of a DelayExpression
     */
    public final String toString(final EsterelModule anEsterelModule) {
        String result = "";
        if (this.anExpression != null) {
            result +=
                this.anExpression.toString(anEsterelModule)
                + " "
                + this.aSignalExpression.toString(anEsterelModule);
        } else if (this.isImmediate && this.withImmediate) {
            result = " # "
                + this.aSignalExpression.toString(anEsterelModule)
                + " ";
        } else {
            result = this.aSignalExpression.toString(anEsterelModule);
        }
        return result;
    }

    }
```

C.2.14. EsterelExpression

Der Klassenaufbau ist in der Abbildung C.2.14 dargestellt.

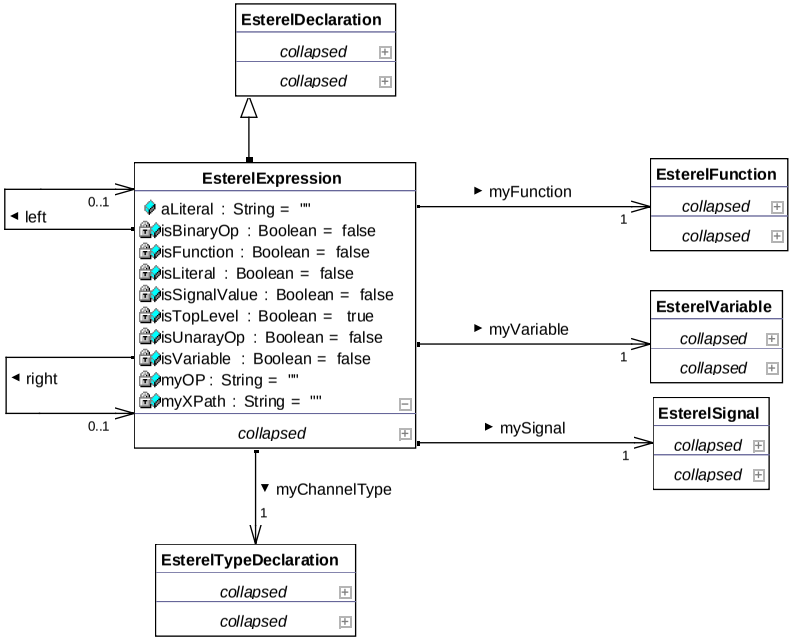


Abbildung C.10.: Klassediagramm `EsterelExpression`

Auflistung C.19: Die Klasse `EsterelExpression`

```
package kiel.fileInterface.esterel.esterel2estudio;

import java.util.ArrayList;
import java.util.List;
import kiel.dataStructure.CompoundLabel;
import kiel.dataStructure.StringLabel;
import kiel.dataStructure.TransitionLabel;
import kiel.dataStructure.boolexp.BooleanExpression;
import kiel.dataStructure.intexp.IntegerExpression;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.CompoundLabelException;
import kiel.util.CompoundLabelParser;
import kiel.util.DOMHelpers;
```

```
import org.jdom.Document;
import org.jdom.Element;

/**
 * <p>
 *  This class implements all esterell expressions.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2004
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.33 $ last modified $Date: 2006/02/13 14:00:04 $
 */
public class EsterelExpression extends EsterelDeclaration {
    /**
     * If the expression is a literals it stored as string.
     */
    private String aLiteral;

    /**
     * true, if the expression is a binary operation.
     */
    private boolean isBinaryOp;

    /**
     * If the expression is a function.
     */
    private boolean isFunction = false;

    /**
     * true, if the expression is a literal.
     */
    private boolean isLiteral;

    /**
     * true, if the expression is a signal value.
     */
    private boolean isSignalValue;
/**
 * <code> true </code>, if this expression is the root expression.
 */
    private boolean isTopLevel = true;

    /**
     * true, if the expression is a unary operation.
     */
    private boolean isUnarayOp;

    /**
     * If the expression is a variable.
     */
    private boolean isVariable;

    /**
     * the left side of a binary operation or the only side of an unary
     * operation.
     */
    private EsterelExpression left;

    /**
     * The id of the function.
     */
    private EsterelFunction myFunction;

    /**
     * the operation as string.
     */
    private String myOP;

    /**
     * the signalid as string.
     */
    private String mySignalId;

    /**
     * the type as string.
     */
    private EsterelTypeDeclaration myType;

    /**
     * The variable.
     */
    private String myVariableId;

    /**
     * the XPath to the expression.
     */
    private String myXPath;

    /**
     * the rigth part of an operation.
     */
    private EsterelExpression right;

    /**
     * the standart constructor.
     */
    public EsterelExpression() {
        super();
        this.isVariable = false;
        this.mySignalId = "";
        this.myXPath = "";
        this.myOP = "";
        this.isUnarayOp = false;
        this.isBinaryOp = false;
        this.left = null;
        this.right = null;
        this.aLiteral = null;
```

```
        this.myType = null;
        this.isSignalValue = false;
        this.isLiteral = false;
    }

    /**
     * Uses the standart constructor and parses an expression. <br>
     * Sideeffects: sets the class variables
     *@param theparser a esterel parser.
     * @param aXPath
     *            the xpath to the expression in an xml document
     * @param xmldoc
     *            the <code>org.jdom.Document</code> representation of an
     *            .exp file
     * @throws EsterelParserException
     *             is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     *
     */
    public EsterelExpression(
            final EsterelParser theparser,
            final String aXPath,
            final Document xmldoc)
        throws EsterelParserException {
        this.isVariable = false;
        this.mySignalId = "";
        this.myXPath = "";
        this.myOP = "";
        this.isUnaryOp = false;
        this.isBinaryOp = false;
        this.left = null;
        this.right = null;
        this.aLiteral = null;
        this.myType = null;
        this.isSignalValue = false;
        this.isLiteral = false;
        if (aXPath != null
            && xmldoc != null) {
            this.parseExpression(theparser,
                aXPath,
                xmldoc);
        }
    }
    /**
     * Uses the standart constructor and parses an expression. <br>
     * Sideeffects: sets the class variables
     *
     *@param theparser a esterel parser.
     * @param aXPath
     *            the xpath to the expression in an xml document
     * @param xmldoc
     *            the <code>org.jdom.Document</code> representation of an
     *            .exp file
     * @param topLevel is the root expression.
     * @throws EsterelParserException
     *             is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     */
    public EsterelExpression(
            final EsterelParser theparser,
            final String aXPath,
            final Document xmldoc,
            final boolean topLevel)
        throws EsterelParserException {
        this.isTopLevel = topLevel;
        this.isVariable = false;
        this.mySignalId = "";
        this.myXPath = "";
        this.myOP = "";
        this.isUnaryOp = false;
        this.isBinaryOp = false;
        this.left = null;
        this.right = null;
        this.aLiteral = null;
        this.myType = null;
        this.isSignalValue = false;
        this.isLiteral = false;
        if (aXPath != null
            && xmldoc != null) {
            this.parseExpression(theparser,
                aXPath,
                xmldoc);
        }
    }

    /**
     * Converts the expression to a
     * <code>kiel.dataStructure.intexp.IntererExpression</code>.
     *
     * @param anEsterelModule
     *            the esterel module
     *
     * @return a <code>kiel.dataStructure.intexpr.IntererExpression</code>
     * @see kiel.dataStructure.TransitionLabel
     */
    public final IntegerExpression convertToKiel(
            final EsterelModule anEsterelModule) {
        CompoundLabelParser.getInstance();
        // Setting StateChart for identifier searching
        CompoundLabelParser.setStateChart(
          anEsterelModule.getEsterelStateChart());
        CompoundLabelParser.setAllLocals(null,
          null);
        String aStringLabel = "{"
            + this.toString(anEsterelModule).trim() + "}";
        IntegerExpression anIntExp = null;
        try {
          anIntExp = CompoundLabelParser.parseIntExpression(aStringLabel);
        } catch (CompoundLabelException ex) {
          anIntExp = null;
        }
        return anIntExp;
```

```
    }

    /**
     * Converts the expression to a
     * <code>kiel.dataStructure.intexp.IntegerExpression</code>.
     *
     * @param anEsterelModule
     *                the esterel module
     * @param theLocalEvents
     *                all local events
     * @param theLocalVariables
     *                all local variables
     * @return a <code>kiel.dataStructure.intexpr.IntererExpression</code>
     * @see kiel.dataStructure.TransitionLabel
     */
    public final IntegerExpression convertToKiel(
           final EsterelModule anEsterelModule,
           final ArrayList theLocalEvents,
           final ArrayList theLocalVariables) {
        CompoundLabelParser.getInstance();
        // Setting StateChart for identifier searching
        CompoundLabelParser.setStateChart(
            anEsterelModule.getEsterelStateChart());
        CompoundLabelParser.setAllLocals(theLocalVariables,
            theLocalEvents);
        String aStringLabel = "{"
                + this.toString(anEsterelModule).trim() + "}";
        IntegerExpression anIntExp = null;
        try {
            anIntExp = CompoundLabelParser.parseIntExpression(aStringLabel);
        } catch (CompoundLabelException ex) {
            anIntExp = null;
        }
        return anIntExp;
    }
    /*
     * Converts the expression to a
     * <code>kiel.dataStructure.floatexp.FloatExpression</code>.
     *
     * @param anEsterelModule
     *                the esterel module
     * @param theLocalEvents
     *                all local events
     * @param theLocalVariables
     *                all local variables
     * @return a <code>kiel.dataStructure.floatexpr.FloatExpression</code>
     * @see kiel.dataStructure.TransitionLabel
     *
    public final FloatExpression convertToKielFloatExp(
           final EsterelModule anEsterelModule,
           final ArrayList theLocalEvents,
           final ArrayList theLocalVariables) {
        CompoundLabelParser.getInstance();
        // Setting StateChart for identifier searching
        CompoundLabelParser.setStateChart(
            anEsterelModule.getEsterelStateChart());
        CompoundLabelParser.setAllLocals(theLocalVariables,
            theLocalEvents);
        String aStringLabel = "{"
                + this.toString(anEsterelModule).trim() + "}";
        FloatExpression aFloatExp = null;
        try {
           aFloatExp = CompoundLabelParser.parseFloatExpression(aStringLabel);
        } catch (CompoundLabelException ex) {
           aFloatExp = null;
        }
        return aFloatExp;
    }

    /**
     * Converts the expression to a
     * <code>kiel.dataStructure.doubleexp.DoubleExpression</code>.
     *
     * @param anEsterelModule
     *                the esterel module
     * @param theLocalEvents
     *                all local events
     * @param theLocalVariables
     *                all local variables
     * @return a <code>kiel.dataStructure.floatexpr.FloatExpression</code>
     * @see kiel.dataStructure.TransitionLabel
     *
    public final DoubleExpression convertToKielDoubleExp(
           final EsterelModule anEsterelModule,
           final ArrayList theLocalEvents,
           final ArrayList theLocalVariables) {
        CompoundLabelParser.getInstance();
        // Setting StateChart for identifier searching
        CompoundLabelParser.setStateChart(
            anEsterelModule.getEsterelStateChart());
        CompoundLabelParser.setAllLocals(theLocalVariables,
            theLocalEvents);
        String aStringLabel = "{"
                + this.toString(anEsterelModule).trim() + "}";
        DoubleExpression aDoubleExp = null;
        try {
            aDoubleExp = CompoundLabelParser
            .parseDoubleExpression(aStringLabel);
        } catch (CompoundLabelException ex) {
            aDoubleExp = null;
        }
        return aDoubleExp;
    }*/
    /**
     * Converts the expression to a
     * <code>kiel.dataStructure.TransitionLabel</code>.
     *
     * @param anEsterelModule
     *                the esterel module
     * @param theLocalEvents
     *                all local events
     * @param theLocalVariables
```

```
     *                all local variables
     * @param aTransLabel
     *                an old label or null
     * @return a <code>kiel.dataStructure.TransitionLabel</code>
     * @see kiel.dataStructure.TransitionLabel
     */
    public final TransitionLabel convertToKiel(
           final EsterelModule anEsterelModule,
           final ArrayList theLocalEvents,
           final ArrayList theLocalVariables,
           final TransitionLabel aTransLabel) {
        CompoundLabelParser.getInstance();
        // Setting StateChart for identifier searching
        CompoundLabelParser.setStateChart(
            anEsterelModule.getEsterelStateChart());
        CompoundLabelParser.setAllLocals(theLocalVariables,
            theLocalEvents);
        String aStringLabel = "{"
                + this.toString(anEsterelModule).trim() + "}";
        TransitionLabel aLabel = null;
        BooleanExpression aBoolExp = null;
        try {
            aBoolExp = CompoundLabelParser.parseBooleanExpression(aStringLabel);
        } catch (CompoundLabelException ex) {
            return aLabel = new StringLabel(
                    this.toString(anEsterelModule).trim());
        }
        if (aTransLabel != null) {
            if (aTransLabel instanceof StringLabel) {
                aLabel = new StringLabel(
                        aStringLabel);
            } else {
                aLabel = aTransLabel;
                ((CompoundLabel) aLabel).setCondition(aBoolExp);
            }
        } else {
            aLabel = new CompoundLabel();
            ((CompoundLabel) aLabel).setCondition(aBoolExp);
        }
        return aLabel;
    } // covert


    /**
     * Converts the expression to a
     * <code>kiel.dataStructure.boolexp.BooleanExpression</code>.
     *
     * @param anEsterelModule
     *                the esterel module
     * @param theLocalEvents
     *                all local events
     * @param theLocalVariables
     *                all local variables
     * @return a <code>kiel.dataStructure.boolexpr.BooleanExpression</code>
     * @see kiel.dataStructure.TransitionLabel
     */
    public final BooleanExpression convertToKielBoolExp(
           final EsterelModule anEsterelModule,
           final ArrayList theLocalEvents,
           final ArrayList theLocalVariables) {
        CompoundLabelParser.getInstance();
        // Setting StateChart for identifier searching
        CompoundLabelParser.setStateChart(
            anEsterelModule.getEsterelStateChart());
        CompoundLabelParser.setAllLocals(theLocalVariables,
            theLocalEvents);
        String aStringLabel = "{"
               + this.toString(anEsterelModule).trim() + "}";
        BooleanExpression aBoolExp = null;
        try {
            aBoolExp = CompoundLabelParser.parseBooleanExpression(aStringLabel);
        } catch (CompoundLabelException ex) {
            aBoolExp = null;
        }
        return aBoolExp;
    }
    /**
     * Converts the expression to a
     * <code>String</code>.
     *
     * @param anEsterelModule
     *                the esterel module
     * @return a <code>kiel.dataStructure.boolexpr.BooleanExpression</code>
     * @see kiel.dataStructure.TransitionLabel
     */
    public final String convertToKielString(
           final EsterelModule anEsterelModule) {
        return this.toString(anEsterelModule);
    }

    /**
     * @return Returns the myType.
     */
    public final EsterelTypeDeclaration getMyType() {
        return this.myType;
    }

    /**
     * Parses a Expression. <br>
     * Sideeffects: sets the class variables.
     *
     *@param theparser a esterel parser.
     * @param aXPath
     *                the XPath which leads to the Expression
     * @param xmldoc
     *                the <code>org.jdom.Document</code> reprsentation of the
     *                .exp file
     * @throws EsterelParserException
     *                is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     */
    public final void parseExpression(
```

```
    final EsterelParser theparser,
    final String aXPath,
    final Document xmldoc)
    throws EsterelParserException {
int i = 0;
this.myXPath = aXPath;
List anElementList = DOMHelpers.getElements(this.myXPath
       + "[not(local-name(.)='EOV')][not(local-name(.)='NULL')]",
    xmldoc);
// the XPath finds the children of the Expression
String actualElement = ((Element) anElementList.get(i))
.getParentElement()
       .getName();
if (actualElement == "UnaryOp") {
    this.isUnarayOp = true;
    this.myType =
        theparser
        .getEsterelModule(theparser.getactualEsterModule())
        .getEsterelTypeByID(
            ((Element) anElementList.get(0)).getAttributeValue("id"));
    this.myOP = this.getOperation(anElementList);
    this.left = this.getExpression(theparser,
        anElementList,
        xmldoc);
} else if (actualElement == "BinaryOp") {
    this.isBinaryOp = true;
    this.myType =
        theparser
        .getEsterelModule(theparser.getactualEsterModule())
        .getEsterelTypeByID(
            ((Element) anElementList.get(0)).getAttributeValue("id"));
    this.myOP = this.getOperation(anElementList);
    this.left = this.getExpression(
        theparser,
        anElementList,
        xmldoc);
    this.right = this.getExpression(
        theparser,
        anElementList,
        xmldoc);
} else if (actualElement == "LoadSignalValueExpression") {
    this.isSignalValue = true;
    this.myType =
        theparser
        .getEsterelModule(theparser.getactualEsterModule())
        .getEsterelTypeByID(
            ((Element) anElementList.get(0)).getAttributeValue("id"));
    this.mySignalId = this.getSignal(anElementList);
} else if (actualElement == "LoadVariableExpression") {
    this.isVariable = true;
    this.myType =
        theparser
        .getEsterelModule(theparser.getactualEsterModule())
        .getEsterelTypeByID(
            ((Element) anElementList.get(0)).getAttributeValue("id"));
    this.myVariableId = this.getVariable(anElementList);
    List anNewElementList = DOMHelpers.getElements(
        "//*[local-name(.)='BuiltinConstantSymbol'][@id='"
        + this.myVariableId
        + "']"
        + "[not(local-name(.)='EOV')][not(local-name(.)='NULL')]",
       xmldoc);
       if (anNewElementList.size() > 0
              && ((Element) anNewElementList.get(0))
              .getName()
              .compareTo("BuiltinConstantSymbol") == 0) {
         this.isVariable = false;
         this.myVariableId = "";
         this.isLiteral = true;
         this.aLiteral = ((Element) anNewElementList.get(0))
         .getChild("S").getText();
                       }
    } else if (actualElement == "Literal") {
        this.isLiteral = true;
        this.myType =
            theparser
            .getEsterelModule(theparser.getactualEsterModule())
            .getEsterelTypeByID(
                ((Element) anElementList.get(0)).getAttributeValue("id"));
        this.aLiteral = ((Element) anElementList.get(1)).getText();
    } else if (actualElement == "FunctionCall" ||
         actualElement.compareTo("BuiltinFunctionSymbol") == 0 ) {
       this.isFunction = true;
       this.myFunction =
          new EsterelFunction(
              theparser,
              "//*[@id='"
              + ((Element) anElementList.get(i))
              .getParentElement()
              .getAttributeValue("id")
              + "' ]/*",
              xmldoc);
  } else {
      throw new EsterelParserException(
           "EsterelExpression: cannot identify expression: "
                  + actualElement);
   }
}
/**
 * @param literal The aLiteral to set.
 */
public final void setALiteral(
       final String literal) {
   this.aLiteral = literal;
}
/**
 * @param theLiteral The isLiteral to set.
 */
public final void setLiteral(
       final boolean theLiteral) {
   this.isLiteral = theLiteral;
}
```

```
    /**
     * @param aType
     *                The myType to set.
     */
    public final void setType(
            final EsterelTypeDeclaration aType) {
        this.myType = aType;
    }

    /**
     * Returns the String representation of the expression.
     *@param anEsterelModule an <code>EsterelModule</code>
     * @return the String representation of an EsterelExpression
     */
    public final String toString(
            final EsterelModule anEsterelModule) {
        String result = "";
        if (this.isBinaryOp) {
            String leftexp = this.left.toString(anEsterelModule);
            String rigthexp = this.right.toString(anEsterelModule);
            if (this.isTopLevel) {
                result = leftexp
                        + " " + this.myOP + " " + rigthexp;
            } else {
                result = "("
                        + leftexp + " " + this.myOP + " " + rigthexp + " )";
            }
        } else if (this.isUnarayOp) {
            String leftexp = this.left.toString(anEsterelModule);
            if (this.isTopLevel) {
                result = this.myOP
                        + " " + leftexp;
            } else {
                result = this.myOP
                        + "( " + leftexp + " )";
            }
        } else if (this.isSignalValue) {
            result = " ? "
                    + anEsterelModule.getEsterelSignalByID(this.mySignalId)
                            .getSignalIdentifier();
        } else if (this.isLiteral) {
            result = this.aLiteral;
        } else if (this.isVariable) {
            EsterelVariable dummy = anEsterelModule
               .getEsterelVariableByID(this.myVariableId);
            if (dummy != null) {
             result = dummy.getVariableIdentifier();
            }  else {
             result = anEsterelModule.getEsterelConstantByID(this.myVariableId)
             .getConstantIdentifier();
            }
        } else if (this.isFunction) {
            if (Esterel2EstudioProperties.getFunctions()) {
            result = anEsterelModule
            .getEsterelFunctionbyID(this.myFunction.getFunctionId())
                    .getFunctionIdentifier();
                } else {
                result = this.myFunction.toString(anEsterelModule);
                }
            } else {
                result = "";
            }
            return result;
        } //toString

        /**
         * Retuns an <code>EsterelSignalExpression</code> of a sub
         * Signalexpession. <br>
         * Sideeffects: none
         *
         *@param theparser a esterel parser.
         * @param anElementList
         *                the cildren elements of the SignalExpression
         * @param xmldoc
         *                the <code>org.jdom.Document</code> reprsentation of the
         *                .exp file
         * @throws EsterelParserException
         *                is only delivered
         * @see kiel.fileInterface.esterel.EsterelParserException
         * @return a sub SignalExpression
         */
        private EsterelExpression getExpression(
                final EsterelParser theparser,
                final List anElementList,
                final Document xmldoc)
                throws EsterelParserException {
            final int firstindex = 2;
            final int secondindex = 3;
            int actualElementNumber = 0;
            if (this.left == null) {
                actualElementNumber = firstindex;
            } else if (this.right == null) {
                actualElementNumber = secondindex;
            } else {
                actualElementNumber = -1;
            }
            if (actualElementNumber == firstindex) {
                return new EsterelExpression(
                    theparser,
                        this.myXPath
                                + "[local-name(.)='"
                                + ((Element) anElementList.get(actualElementNumber))
                                .getName()
                                + "'][@id='"
                                + ((Element) anElementList.get(actualElementNumber))
                                .getAttributeValue("id")
                                + "']/*", xmldoc, false);
            } else if (actualElementNumber == secondindex) {
                return new EsterelExpression(
                    theparser,
                        this.myXPath
```

```
                    + "[local-name(.)='"
                    + ((Element) anElementList.get(actualElementNumber))
                    .getName()
                    + "'][@id='"
                    + ((Element) anElementList.get(actualElementNumber))
                    .getAttributeValue("id")
                    + "']/*", xmldoc, false);
        } else {
            return null;
        }
    }
    /**
     * Retuns the operation of the expession. <br>
     * Sideeffects: none
     *
     * @param anElementList
     *                the cildren elements of the Expression
     * @return the operations name
     * @throws EsterelParserException
     *                if no operation is found
     */
    private String getOperation(
            final List anElementList)
            throws EsterelParserException {
        if (((Element) anElementList.get(1)).getName() == "S") {
            String op = ((Element) anElementList.get(1)).getText();
            if (op.compareTo("&lt;") == 0) {
                return "<";
            } else if (op.compareTo("&lt;=") == 0) {
                return "<=";
            } else if (op.compareTo("&gt;") == 0) {
                return ">";
            } else if (op.compareTo("&gt;=") == 0) {
                return ">=";
            } else if (op.compareTo("&lt;&gt;") == 0) {
                return "<>";
            } else {
                return op;
            }
        }
        throw new EsterelParserException(
                "EsterelEspression found no operation");
    }

    /**
     * Retuns the id of the function. <br>
     * Sideeffects: none
     *
     * @param anElementList
     *                the cildren elements of the SignalExpression
     * @return the signals id
     * @throws EsterelParserException
     *                if no signal is found
     */
    private String getSignal(
            final List anElementList)
            throws EsterelParserException {
        final int theSignalIndex = 1;
        if (anElementList.size() > theSignalIndex) {
            return ((Element) anElementList.get(theSignalIndex))
            .getAttributeValue("id");
        }
        throw new EsterelParserException(
                "EsterelExpression : found no signal ");
    }

    /**
     * Retuns the type of the Signalexpession. <br>
     * Sideeffects: none
     *
     * @param anElementList
     *                the cildren elements of the SignalExpression
     * @return the type name
     * @throws EsterelParserException
     *                if no variable is found
     */
    private String getVariable(
            final List anElementList)
            throws EsterelParserException {
        final int theVariableIndex = 1;
        if (anElementList.size() > theVariableIndex) {
            return ((Element) anElementList.get(theVariableIndex))
            .getAttributeValue("id");
        }
        throw new EsterelParserException(
                "EsterelExpression: no Variable found.");
    }
}
```

C.2.15. EsterelFunctionDeclaration

Der Klassenaufbau ist in der Abbildung C.2.15 dargestellt.

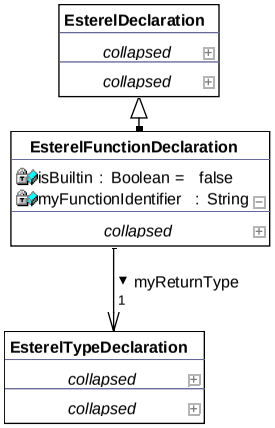


Abbildung C.11.: Klassediagramm `EsterelFunctionDeclaration`

Auflistung C.20: Die Klasse `EsterelFunctionDeclaration`

```
package kiel.fileInterface.esterel.esterel2estudio;
import java.util.ArrayList;
import java.util.List;

import kiel.dataStructure.Variable;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;

import org.jdom.Element;
/**
 * <p> It represents esterel Functions .</p>
 *  <p>Copyright: Copyright (c) 2005</p>
 * <p>Company: Uni Kiel</p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
 * @version $Revision: 1.26 $ last modified $Date: 2006/02/06 18:35:29 $
 * <br>
 */
public class EsterelFunctionDeclaration
 extends EsterelDeclaration {
/**
 * A function will be realized as Variable.
 */
private EsterelVariable inKiel = null;
    /**
     * true, if the function is a built in function, else false.
     */
        private boolean isBuiltin = false;
    /**
     * the Function name.
     */
    private String myFunctionIdentifier;
    /**
     * the return type.
     */
    private EsterelTypeDeclaration myReturnType;
    /**
```

```
 * the value parameters.
 */
private ArrayList myValueParameterTyps;
/**
 * the simple constructor.
 *
 */
 public EsterelFunctionDeclaration() {
    super();
    this.myReturnType = null;
    this.myValueParameterTyps = null;
    this.myFunctionIdentifier = "";
}
/**
 * Creates an EsterelFunction and fills the class variables.
 * <br>Sideeffects: class variables are changed.
 * @param theparser the esterel parser
 * @param theXPath the XPath to the EsterelVariable
 * @throws EsterelParserException if something
 * goes wrong at the parsing.
 */
public EsterelFunctionDeclaration(
    final EsterelParser theparser,
    final String theXPath)
    throws EsterelParserException {
    super();
    this.myReturnType = null;
    this.myValueParameterTyps = null;
    this.myFunctionIdentifier = "";
    if (theXPath != null && theparser != null) {
        this.parseEsterelStatement(theparser, theXPath);
    } else {
        throw new EsterelParserException(
      "FunctionDelclaration : no parser or path");
    }
}
 /**
  * Returns a variable representation of the esterel variable.
  * @param anEsterelModule the esterel module which
  * becomes a KIEL Statechart
  * @param theLocalEvents Stack with the local events
  * @param theLocalVariables Stack with all local variables
  * @throws Esterel2EstudioException
  *                 if the convertion to Kiel is not working. <br>
  *                 Sideeffects: none
  * @return a State
  * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
  * @see kiel.dataStructure.State
  */
 public final Variable convertToKiel(
        final EsterelModule anEsterelModule,
        final ArrayList theLocalEvents,
        final ArrayList theLocalVariables)
       throws Esterel2EstudioException {
    if (this.inKiel == null) {
        this.inKiel = new EsterelVariable();
        this.inKiel.setVariableIdentifier(this.myFunctionIdentifier);
        this.inKiel.setNotinEsterelProgram(true);
        this.inKiel.setChannelType(this.myReturnType);
    }
    return this.inKiel
    .convertToKiel(anEsterelModule,
           theLocalEvents, theLocalVariables);
}
/**
*
* @return the function identifier.
*/
   public final String getFunctionIdentifier() {
        return this.myFunctionIdentifier;
   }

/**
 * @return Returns the isBuiltin.
 */
public final boolean isBuiltin() {
   return this.isBuiltin;
}
/**
  * Parses a Function.
  * <br> Sideeffects: sets the class variables.
  * @param theXPath the XPath which leads to the Expression
  * @param theparser the esterel parser
  * @throws EsterelParserException if there is a parsing problem
  * @see kiel.fileInterface.esterel.EsterelParserException
  */
public final void parseEsterelStatement(
    final EsterelParser theparser,
    final String theXPath)
    throws EsterelParserException {
    List anElementList =
        DOMHelpers.getElements(theXPath, theparser.getXMLDocument());

    int theEOVCounter = 0;
    int i = 0;
    this.setName(((Element) anElementList.get(0)).getParentElement()
           .getName());
    this.setXMLID(((Element) anElementList.get(0)).getParentElement()
           .getAttributeValue("id"));
    this.myReturnType = null;
    if (this.getName().compareTo("BuiltinFunctionSymbol") == 0) {
        this.isBuiltin = true;
        this.myFunctionIdentifier =
            ((Element) anElementList.get(0)).getText();
    } else {
    this.myValueParameterTyps = new ArrayList();
    while (i < anElementList.size()) {
        Element anElement = (Element) anElementList.get(i);
        if (anElement.getName().compareTo("EOV") == 0) {
            theEOVCounter++;
        } else if (i == 0) { // parse name}
            this.myFunctionIdentifier = anElement.getText();
```

```
    } else if (theEOVCounter == 0) {
        this.myValueParameterTyps.add(
            theparser
                .getEsterelModule(theparser.getactualEsterModule())
                .getEsterelTypeByID(((Element) anElementList.get(i))
                        .getAttributeValue("id")));
    } else if (theEOVCounter == 1) {
        this.myReturnType =
            theparser
                .getEsterelModule(theparser.getactualEsterModule())
                .getEsterelTypeByID(((Element) anElementList.get(i))
                        .getAttributeValue("id"));
        }
        i++;
    } //while
    }
}
/**
 * Returns the String representation of the variable.
 * @param anEsterelModule an <code>EsterelModule</code>
 * @return the String representation of an EsterelVariable
 */
public final String toString(
        final EsterelModule anEsterelModule) {
    if (this.isBuiltin) {
        return "";
        }
    String result = this.myFunctionIdentifier + " ( ";
    if (this.myValueParameterTyps != null) {
        for (int i = 0; i < this.myValueParameterTyps.size(); i++) {
            result += (String) this.myValueParameterTyps.get(i)
                + " ,";
            }
        }
    result += " ) : " + this.myReturnType.getTypeName();
    return result;
    }
}
```

C.2.16. EsterelFunction

Auflistung C.21: Die Klasse `EsterelFunction`

```
package kiel.fileInterface.esterel.esterel2estudio;

import java.util.List;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import org.jdom.Document;
import org.jdom.Element;

/**
 * <p>
 *  Represents a function call in esterel.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.6 $ last modified $Date: 2006/02/07 18:39:37 $
 */
public final class EsterelFunction
       extends EsterelDeclaration {
    /**
     * the functionexpressions.
     */
   private EsterelExpression[] myExpressions;
   /**
    * the xml id of the functiondelaration.
    */
   private String myFunctionXMLID;
/**
 * the function identifier.
 */
   private String myFunctionId;
/**
 * the return type.
 */
   private EsterelTypeDeclaration myReturnType;
/**
 * a constructor.
 * @param theparser a EsterelParser.
 * @param aXpath the path to the function call
 * @param xmldoc the document
 * @throws EsterelParserException an exception
 */
   public EsterelFunction(
           final EsterelParser theparser,
           final String aXpath,
           final Document xmldoc)
           throws EsterelParserException {
       super();
       this.parseEsterelStatement(
           theparser,
           aXpath,
           xmldoc);
   }

   /**
    * @return Returns the myFunctionId.
    */
   public String getFunctionId() {
       return this.myFunctionId;
   }

   /**
    * parses a function into the datastructure.
    *
    *@param theparser a EsterelParser.
    * @param aXpath
    *                the xpath to the function.
    * @param xmldoc
    *                the document
    * @throws EsterelParserException
    *                an exception
    */
   public void parseEsterelStatement(
           final EsterelParser theparser,
           final String aXpath,
           final Document xmldoc)
           throws EsterelParserException {
       List anElementList = DOMHelpers.getElements(aXpath,
           xmldoc);
       final int aTHREE = 3;
       if (anElementList.size() > 1) {
           if ((((Element) anElementList.get(0))).getName()
                   .compareTo("NULL") != 0) {
               this.myReturnType =
                   theparser
                   .getEsterelModule(theparser.getactualEsterModule())
                   .getEsterelTypeByID(((Element) anElementList.get(0))
                       .getAttributeValue("id"));
           }
           if ((((Element) anElementList.get(1))).getName()
                   .compareTo("NULL") != 0) {
               this.myFunctionXMLID =
                   ((Element) anElementList.get(1)).getAttributeValue("id");
               this.myFunctionId =
                   theparser.getEsterelModule(theparser
                           .getactualEsterModule())
                   .getEsterelFunctionbyID(this.myFunctionXMLID)
```

```
        .getFunctionIdentifier();
    }
    if (anElementList.size() > aTHREE) {
        this.myExpressions =
            new EsterelExpression[anElementList.size() - aTHREE];
        for (int i = 2; i < anElementList.size() - 1; i++) {
            this.myExpressions[i - 2] =
                new EsterelExpression(
                    theparser,
                    "//*[@id='"
                        + ((Element) anElementList.get(i))
                            .getAttributeValue("id")
                        + "']/*",
                    xmldoc);
        }
    }
} else {
    throw new EsterelParserException(
        "EsterelFunction1: missing elements. ");
    }
}

/**
 * @param anEsterelModule
 *            the module.
 * @return a string.
 */
public String toString(
    final EsterelModule anEsterelModule) {
    String result = "";
    EsterelFunctionDeclaration theFDecl = anEsterelModule
    .getEsterelFunctionbyID(this.myFunctionXMLID);
    if (theFDecl.isBuiltin()) {
        if (this.myExpressions.length == 1) {
            result = theFDecl.getFunctionIdentifier()
                + " "
                + this.myExpressions[0].toString(anEsterelModule);
        } else {
            result = this.myExpressions[0].toString(anEsterelModule)
                + " "
                + theFDecl.getFunctionIdentifier()
                + " "
                + this.myExpressions[1].toString(anEsterelModule);
        }
    } else {
        result = theFDecl.getFunctionIdentifier();
        if (!Esterel2EstudioProperties.getFunctions()) {
            result += "( ";
            if (this.myExpressions != null) {
                for (int i = 0; i < this.myExpressions.length - 1; i++) {
                    result += this.myExpressions[i]
                            .toString(anEsterelModule)
                        + Esterel2EstudioProperties
                            .getSeperatorString();
                }
                if (this.myExpressions.length > 1) {
                    result +=
                        this.myExpressions[this.myExpressions.length - 1]
                                .toString(anEsterelModule);
                }
            }
            result += ")";
        }
        return result;
    }
}
```

C.2.17. EsterelModule

Der Klassenaufbau ist in der Abbildung C.2.17 dargestellt.

Auflistung C.22: Die Klasse `EsterelModule`

```
package kiel.fileInterface.esterel.esterel2estudio;

//
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.ORState;
import kiel.dataStructure.State;
import kiel.dataStructure.StateChart;
import kiel.dataStructure.Variable;
import kiel.dataStructure.eventexp.Event;
import kiel.dataStructure.eventexp.Signal;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StateChartHelpers;
import org.jdom.Element;

/**
 * <p>
 *  This class implements an esterel module.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2004
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.53 $ last modified $Date: 2006/02/13 14:00:04 $
 */
public class EsterelModule
        extends EsterelDeclaration {
    /** the esterel statements in that module. */
    private ArrayList anEsterelProgram = new ArrayList();
    /** the statechart.
     *
     */
    private StateChart anEsterelStateChart = null;
    /**
     * the module constants.
     */
    private EsterelConstant[] myConstants = null;

    /**
     * the module functions.
     */
    private EsterelFunctionDeclaration[] myFunctions = null;

    /** the input and output signals . */
    private EsterelSignal[] myInputOutputSignals = null;

    /** the input signals. */
    private EsterelSignal[] myInputSignals = null;

    /** the output signals. */
    private EsterelSignal[] myOutputSignals = null;

    /**
     * the module procedures.
     */
    private EsterelProcedureDeclaration[] myProcedures = null;

    /**
     * the module relations.
     */
    private EsterelRelation[] myRelations = null;

    /**
     * the module returns.
     */
     private EsterelSignal[] myReturnSignals = null;

    /**
     * the module sensors.
     */
    private EsterelSignal[] mySensors = null;

    /**
     * the module tasks.
     */
    private EsterelTaskDeclaration[] myTasks = null;

    /**
     * the module typs.
     */
    private EsterelTypeDeclaration[] myTyps = null;

    /**
     * XPath which leads to a module.
     */
    private String myXPath;

    /** A representation of all esterel signals. */
```

```
    private EsterelSignal[] theEsterelSignals = null;

100 /** A representation of all esterel variables. */
    private EsterelVariable[] theEsterelVariables = null;

    //public final void parseEsterelModule(final EsterelParser the
    // parser){
    /**
     * Simple constructor.
     */
    public EsterelModule() {
        this.myXPath = "";
110 } //public EsterelModule()

    /**
     * Parses a Document and set the class varaibles.
     *
     * @param myModuleXPath
     *                the xpath to the module
     * @throws EsterelParserException
     *                 if there are parsing errors
     * @see #parseEsterelModule(EsterelParser)
120  * @see kiel.fileInterface.esterel.EsterelParser
     */
    public EsterelModule(
           final String myModuleXPath)
           throws EsterelParserException {
       this();
       this.myXPath = myModuleXPath
              + "/*[local-name(.)='Module']//*";
    } //public EsterelModule(final EsterelParser theparser) {

130 /**
     * Simple constructor that sets the class variable
     * <code>aModuleName</code >
     * and XPath to the module.
     * @param name a name
     * @param myModuleXPath the Xpath to the module.
     */
    public EsterelModule(final String name,
           final String myModuleXPath) {
       this.setName(name);
140    this.myXPath = myModuleXPath
              + "/*[local-name(.)='Module']/*";
    } //public EsterelModule()
  /**
   * adds a new signal to an <code>EsterelModule</code>.
   * @return the <code>EsterelSignal</code>
   */
    public final EsterelSignal addNewSignal() {
        EsterelSignal result = new EsterelSignal();
        if (this.theEsterelSignals.length > 0) {
150         result.setXMLID(this.theEsterelSignals[
                    this.theEsterelSignals.length - 1]
                                          .getXMLID()
                + "1");
        } else {
            result.setXMLID("1");
        }
        result.setName("EsterelSignal");
        // Array to ArrayList
        ArrayList dummy = new ArrayList();
160     for (int i = 0; i < this.theEsterelSignals.length; i++) {
            dummy.add(this.theEsterelSignals[i]);
        }
        dummy.add(result);
        this.theEsterelSignals = new EsterelSignal[dummy.size()];
        for (int i = 0; i < dummy.size(); i++) {
            this.theEsterelSignals[i] = (EsterelSignal) dummy.get(i);
        }
        Arrays.sort(this.theEsterelSignals,
            new EsterelDeclarationSortByID());
170     return result;
    }
    /**
     * adds a new signal to an <code>EsterelModule</code>.
     * @return the <code>EsterelSignal</code>
     */
    public final EsterelVariable addNewVariable() {
        EsterelVariable result = new EsterelVariable();
        if (this.theEsterelVariables.length > 0) {
            result.
180         setXMLID(this.theEsterelVariables[
                        this.theEsterelVariables.length - 1]
                                      .getXMLID()
                    + "1");
        } else {
            result.setXMLID("1");
        }
        result.setName("EsterelVariable");
        // Array to ArrayList
        ArrayList dummy = new ArrayList();
190     for (int i = 0; i < this.theEsterelVariables.length; i++) {
            dummy.add(this.theEsterelVariables[i]);
        }
        dummy.add(result);
        this.theEsterelVariables = new EsterelVariable[dummy.size()];
        for (int i = 0; i < dummy.size(); i++) {
            this.theEsterelVariables[i] = (EsterelVariable) dummy.get(i);
        }
        Arrays.sort(this.theEsterelVariables,
            new EsterelDeclarationSortByID());
200     return result;
    }

    /**
     * Adds a statement to the esterel module at index i.
     *
     * @param index
     *                the index of the statement
     * @param aStatement
     *                an <code>EsterelStatement</code> subclass <br>
```

```
 *                Sideeffects: the class variable
 *                <code>anEsterelProgram</code> is changed
 * @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
 */
public final void addStatementToModule(
       final int index,
       final EsterelStatement aStatement) {
   this.anEsterelProgram.add(index,
       aStatement);
} //public final void addStatementToModule(


/**
 * Returns a KIEL statechart representation of the esterel module. <br>
 * Sideeffects: none
 *
 * @return a State
 * @throws Esterel2EstudioException
 *                if there is an exception in the convertion.
 * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
 * @see kiel.dataStructure.State
 */
public final StateChart convertToKielChart()
       throws Esterel2EstudioException {
   this.anEsterelStateChart = new StateChart();
    ORState root = new ORState();
   InitialState anInitialState = new InitialState();
   InitialArc aInitialArc = new InitialArc();
   //add types not implemented yet
   //add constants
   ArrayList theLocalVariables = new ArrayList();
   Variable[] theConstants = new Variable[this.myConstants.length];
   for (int i = 0; i < this.myConstants.length; i++) {
       theConstants[i] = this.myConstants[i].
       convertToKiel(this, null, null);
   }
   StateChartHelpers.addLocalVars(root,
       theConstants);
   //add functions as Variable
   if (Esterel2EstudioProperties.getFunctions()) {
   Variable[] theFunctions = new Variable[this.myFunctions.length];
   for (int i = 0; i < this.myFunctions.length; i++) {
       theFunctions[i] =
           this.myFunctions[i].convertToKiel(this, null, null);
   }
   StateChartHelpers.addLocalVars(root,
       theFunctions);
   }
   int oldsize = theLocalVariables.size();
   theLocalVariables.addAll(root.getVariables());
   // add signals
   Event[] theInputEvents = new Event[this.myInputSignals.length
                                      + this.myInputOutputSignals.length
                                      + this.myReturnSignals.length
                                      + this.mySensors.length];
   Event[] theOutputEvents = new Event[this.myOutputSignals.length ];
   for (int i = 0; i < this.myInputSignals.length; i++) {
       Signal x = this.myInputSignals[i].convertToKiel(this);
       theInputEvents[i] = x;
   }
   for (int i = this.myInputSignals.length;
   i < this.myInputSignals.length
          + this.myInputOutputSignals.length; i++) {
       Signal x =
           this.myInputOutputSignals[i - this.myInputSignals.length]
                                        .convertToKiel(this);
       theInputEvents[i] = x;
   }
   for (int i = this.myInputSignals.length
          + this.myInputOutputSignals.length;
   i < this.myInputSignals.length
          + this
          .myInputOutputSignals.length
          + this.myReturnSignals.length; i++) {
       Signal x =
           this.myReturnSignals[i
                                - this.myInputSignals.length
                                - this.myInputOutputSignals.length]
                                .convertToKiel(this);
       theInputEvents[i] = x;
   }

   for (int i = this.myInputSignals.length
          + this.myInputOutputSignals.length
          + this.myReturnSignals.length;
   i < this.myInputSignals.length
          + this.myInputOutputSignals.length
          + this.myReturnSignals.length
          + this.mySensors.length; i++) {
       Signal x =
           this.mySensors[i
                                - this.myInputSignals.length
                                - this.myInputOutputSignals.length
                                - this.myReturnSignals.length]
                                .convertToKiel(this);
       theInputEvents[i] = x;
   }
   for (int i = 0; i < this.myOutputSignals.length; i++) {
       Signal x = this.myOutputSignals[i].convertToKiel(this);
       theOutputEvents[i] = x;
   }


   this.anEsterelStateChart.setModelSource("Esterel Studio");
   this.anEsterelStateChart.setModelVersion("5.0");
   this.anEsterelStateChart.setInputEvents(theInputEvents);
   this.anEsterelStateChart.setOutputEvents(theOutputEvents);
   State aState = ((EsterelStatement) (this.anEsterelProgram.get(0)))
   .convertToKiel(this,
       true,
       new ArrayList(),
       theLocalVariables,
```

```
            new ArrayList());
        // in this case false means that the return state is not final
        // a new ArrayList for the local events
        // remove the local variables
        while (theLocalVariables.size() != oldsize) {
            theLocalVariables.remove(theLocalVariables.size() - 1);
        }
        aInitialArc.setSource(anInitialState);
        aInitialArc.setTarget(aState);
        root.setName("Module_"
                + this.getName());
        root.addSubnode(anInitialState);
        root.addSubnode(aState);
        this.anEsterelStateChart.setRootNode(root);
        aState.setParent(root);
        anInitialState.setParent(root);
        return this.anEsterelStateChart;
    } //public final StateChart convertToKielChart() {
    /**
     * @param anID
     *                the xmlid.
     * @return the EsterelConstant with the id anID or null.
     */
    public final EsterelConstant getEsterelConstantByID(
            final String anID) {
        for (int i = 0; i < this.myConstants.length; i++) {
            if (this.myConstants[i].getXMLID().compareTo(anID) == 0) {
                return   this.myConstants[i];
            }
        }
        return null;
    }
       /**
       *
       * @param anID a function name.
       * @return <code>EsterelFunctionDelaration</code>
       */
          public final EsterelFunctionDeclaration getEsterelFunctionbyID(
                  final String anID) {
              EsterelFunctionDeclaration result = null;
              for (int i = 0; i < this.myFunctions.length
                      && result == null; i++) {
                  if (this.myFunctions[i].getXMLID()
                          .compareTo(anID) == 0) {
                      result = this.myFunctions[i];
                  }
              }
              return result;
          }

    /**
     * Returns the actual <code>anEsterelProgramm</code>.<br>
     * Sideeffects: none
     *
     * @return <code>anEsterelProgramm</code>
     */
    public final ArrayList getEsterelProgram() {
        return this.anEsterelProgram;
    } //public final ArrayList getEsterelProgram()

    /**
     * @param anID
     *                the xmlid.
     * @return the EsterelProcedureDeclarationl with the id anID or null.
     */
    public final EsterelProcedureDeclaration getEsterelProcedureByID(
            final String anID) {
        EsterelProcedureDeclaration dummy = new EsterelProcedureDeclaration();
        dummy.setXMLID(anID);
        int key = Arrays.binarySearch(this.myProcedures,
            dummy,
            new EsterelDeclarationSortByID());
        if (key < 0) {
            return null;
        }
        return this.myProcedures[key];
    }

    /**
     * @param anID
     *                the xmlid.
     * @return the EsterelTypeDeclarationl with the id anID or null.
     */
    public final EsterelTypeDeclaration getEsterelTypeByID(
            final String anID) {
        EsterelTypeDeclaration dummy =
            new EsterelTypeDeclaration();
        dummy.setXMLID(anID);
        int key = Arrays.binarySearch(this.myTyps,
            dummy,
            new EsterelDeclarationSortByID());
        if (key < 0) {
            return null;
        }
        return this.myTyps[key];
    }
    /**
    *
    * @param aName a type name.
    * @return <code>EsterelTypeDelaration</code>
    */
       public final EsterelTypeDeclaration getEsterelTypeByName(
               final String aName) {
           EsterelTypeDeclaration result = null;
           for (int i = 0; i < this.myTyps.length
                   && result == null; i++) {
               if (this.myTyps[i].getTypeName().compareTo(aName) == 0) {
                   result = this.myTyps[i];
               }
           }
           return result;
       }
```

```
    /**
     * @param anID          the xmlid.
     *
     * @return the EsterelRelationDeclarationl with the id anID or null.
     */
    public final EsterelRelation getEsterelRelationByID(
            final String anID) {
        EsterelRelation dummy = new EsterelRelation();
        dummy.setXMLID(anID);
        int key = Arrays.binarySearch(this.myRelations,
                dummy,
                new EsterelDeclarationSortByID());
        if (key < 0) {
            return null;
        }
        return this.myRelations[key];
    }

    /**
     * @param anID          the xmlid.
     *
     * @return the EsterelTaskDeclarationl with the id anID or null.
     */
    public final EsterelTaskDeclaration getEsterelTaskByID(
            final String anID) {
        EsterelTaskDeclaration dummy = new EsterelTaskDeclaration();
        dummy.setXMLID(anID);
        int key = Arrays.binarySearch(this.myTasks,
                dummy,
                new EsterelDeclarationSortByID());
        if (key < 0) {
            return null;
        }
        return this.myTasks[key];
    }

    /**
     * @param anID          the xmlid.
     *
     * @return the EsterelSignal with the id anID or null.
     */
    public final EsterelSignal getEsterelSignalByID(
            final String anID) {
        EsterelSignal dummy = new EsterelSignal();
        dummy.setXMLID(anID);
        int key = Arrays.binarySearch(this.theEsterelSignals,
                dummy,
                new EsterelDeclarationSortByID());
        if (key < 0) {
            return null;
        }
        return this.theEsterelSignals[key];
    }
    /**
     *
     * @param aName a signal name.
     * @return <code>EsterelSignal</code>
     */
        public final EsterelSignal getEsterelSignalByOrginalName(
                final String aName) {
            EsterelSignal result = null;
            for (int i = 0; i < this.theEsterelSignals.length
                    && result == null; i++) {
                if (this.theEsterelSignals[i].getOrginalSignalIdentifier()
                        .compareTo(aName) == 0) {
                    result = this.theEsterelSignals[i];
                }
            }
            return result;
        }

        /**
         * @param anID          the xmlid.
         *
         * @return the EsterelSignal with the id anID or null.
         */
        public final EsterelVariable getEsterelVariableByID(
                final String anID) {
            EsterelVariable dummy = new EsterelVariable();
            dummy.setXMLID(anID);
            int key = Arrays.binarySearch(this.theEsterelVariables,
                    dummy,
                    new EsterelDeclarationSortByID());
            if (key < 0) {
                return null;
            }
            return this.theEsterelVariables[key];
        }
        /**
         *
         * @param aName a name of a variable
         * @return <code>EsterelVariable</code>
         */
        public final EsterelVariable getEsterelVariableByOrginalName(
                final String aName) {
            EsterelVariable result = null;
            for (int i = 0; i < this.theEsterelVariables.length
                    && result == null; i++) {
                if (this.theEsterelVariables[i].getOrginalVariableIdentifier()
                        .compareTo(aName) == 0) {
                    result = this.theEsterelVariables[i];
                }
            }
            return result;
        }
        /**
         * Parses an esterel module and set values. <br>
         * Sideeffects: see at subclasses
         *
         * @param theparser
         *
```

```
     *                an intance of a EsterelParser
     * @throws EsterelParserException
     *                if there is anything wrong at the parsing
     * @see kiel.fileInterface.esterel.EsterelParser
     */
    public final void parseEsterelModule(
            final EsterelParser theparser)
            throws EsterelParserException {
        //get ModuleName
        List anElementList = DOMHelpers.getElements(this.myXPath,
            theparser.getXMLDocument());
        String aXPath = "//*[@id='"
                + ((Element) anElementList.get(0)).getAttributeValue("id")
                + "']/*[local-name(.)='S']";
        anElementList = DOMHelpers.getElements(aXPath,
            theparser.getXMLDocument());
        this.setName(((Element) anElementList.get(0)).getText());
        //getType
        aXPath = this.myXPath
                + "[local-name(.)='SymbolTable']/*"
                + "[local-name(.)='TypeSymbol' "
                + "or local-name(.)='BuiltinTypeSymbol']/*";
        anElementList = DOMHelpers.getElements(aXPath,
            theparser.getXMLDocument());
        ArrayList theTyps = new ArrayList();
        for (int i = 0; anElementList.size() > i; i++) {
            Element anElement = ((Element) anElementList.get(i));
            EsterelTypeDeclaration aType =
                new EsterelTypeDeclaration(
                    anElement.getName(),
                    anElement.getParentElement().getAttributeValue("id"));
            aType.setTypeName(anElement.getText());
            theTyps.add(aType);
        }
        theTyps.trimToSize();
        this.myTyps =
            (EsterelTypeDeclaration[])
            theTyps.toArray(new EsterelTypeDeclaration[theTyps.size()]);
        Arrays.sort(this.myTyps,
            new EsterelDeclarationSortByID());
        // get functions
        aXPath = this.myXPath
                + "[local-name(.)='SymbolTable']/*"
                + "[local-name(.)='FunctionSymbol' "
                + "or local-name(.)='BuiltinFunctionSymbol']";
        anElementList = DOMHelpers.getElements(aXPath,
            theparser.getXMLDocument());
        ArrayList theFunctions = new ArrayList();
        for (int i = 0; i < anElementList.size(); i++) {
            theFunctions.add(new EsterelFunctionDeclaration(
                    theparser,
                    "//*[@id='"
                            + ((Element) anElementList.get(i))
                            .getAttributeValue("id")
                            + "']/*"));
        } //for
        theFunctions.trimToSize();
        this.myFunctions =
            (EsterelFunctionDeclaration[]) theFunctions
            .toArray(new EsterelFunctionDeclaration[theFunctions.size() ]);
        Arrays.sort(this.myFunctions,
            new EsterelDeclarationSortByID());
        //get constants
        aXPath = this.myXPath
                + "[local-name(.)='SymbolTable']/*"
                + "[local-name(.)='ConstantSymbol']";
        anElementList = DOMHelpers.getElements(aXPath,
            theparser.getXMLDocument());
        ArrayList theConstants = new ArrayList();
        for (int i = 0; anElementList.size() > i; i++) {
            theConstants.add(new EsterelConstant(
                    theparser,
                    "//*[@id='"
                            + ((Element) anElementList.get(i))
                            .getAttributeValue("id")
                            + "']/*"));
        } //for
        theConstants.trimToSize();
        this.myConstants =
            (EsterelConstant[]) theConstants
            .toArray(new EsterelConstant[theConstants.size()]);
        //get procedures
        aXPath = this.myXPath
                + "[local-name(.)='SymbolTable']/*"
                + "[local-name(.)='ProcedureSymbol']";
        anElementList = DOMHelpers.getElements(aXPath,
            theparser.getXMLDocument());
        ArrayList theProcedures = new ArrayList();
        for (int i = 0; i < anElementList.size(); i++) {
            theProcedures.add(new EsterelProcedureDeclaration(
                    theparser,
                    "//*[@id='"
                            + ((Element) anElementList.get(i))
                            .getAttributeValue("id")
                            + "']/*"));
        } //for
        theProcedures.trimToSize();
        this.myProcedures =
            (EsterelProcedureDeclaration[]) theProcedures
            .toArray(new EsterelProcedureDeclaration[theProcedures.size()]);
        Arrays.sort(this.myProcedures,
            new EsterelDeclarationSortByID());
        // get tasks
        aXPath = this.myXPath
                + "[local-name(.)='SymbolTable']/*"
                + "[local-name(.)='TaskSymbol']";
        anElementList = DOMHelpers.getElements(aXPath,
            theparser.getXMLDocument());
        ArrayList theTasks = new ArrayList();
        for (int i = 0; i < anElementList.size(); i++) {
            theTasks.add(new EsterelTaskDeclaration(
                    theparser,
```

```
            "//*[@id='"
                  + ((Element) anElementList.get(i))
                  .getAttributeValue("id")
                  + "']/*"));
} //for
theTasks.trimToSize();
this.myTasks =
    (EsterelTaskDeclaration[]) theTasks
    .toArray(new EsterelTaskDeclaration[theTasks.size()]);
Arrays.sort(this.myTasks,
    new EsterelDeclarationSortByID());
// get signals, returns and sensor and set them
ArrayList theInputSignals = new ArrayList();
ArrayList theOutputSignals = new ArrayList();
ArrayList theReturnSignals = new ArrayList();
ArrayList theSensors = new ArrayList();
ArrayList theInputOutputSignals = new ArrayList();
List allSignals = DOMHelpers
.getElements(this.myXPath +
       "[local-name(.)='SignalSymbol']",
    theparser.getXMLDocument());
allSignals.addAll(DOMHelpers
    .getElements(this.myXPath +
           "[local-name(.)='BuiltinSignalSymbol']",
    theparser.getXMLDocument()));
List allVariables = DOMHelpers
.getElements(this.myXPath +
       "[local-name(.)='VariableSymbol']",
    theparser.getXMLDocument());
this.theEsterelSignals = new EsterelSignal[allSignals.size()];
for (int i = 0; i < allSignals.size(); i++) {
    this.theEsterelSignals[i] = new EsterelSignal(
           theparser, (Element) allSignals.get(i));
}
Arrays.sort(this.theEsterelSignals,
    new EsterelDeclarationSortByID());
this.theEsterelVariables = new EsterelVariable[allVariables.size()];
for (int i = 0; i < allVariables.size(); i++) {
    this.theEsterelVariables[i] = new EsterelVariable(
           theparser, (Element) allVariables.get(i));
}
Arrays.sort(this.theEsterelVariables,
    new EsterelDeclarationSortByID());

for (int i = 0; i < this.theEsterelSignals.length; i++) {
    EsterelSignal anEsterelSignal = this.theEsterelSignals[i];
    if (anEsterelSignal.isInput()) {
        theInputSignals.add(anEsterelSignal);
    } else if (anEsterelSignal.isOutput()) {
        theOutputSignals.add(anEsterelSignal);
    } else if (anEsterelSignal.isReturn()) {
        theReturnSignals.add(anEsterelSignal);
    } else if (anEsterelSignal.isSensor()) {
        theSensors.add(anEsterelSignal);
    } else if (anEsterelSignal.isInputOutput()) {
        theInputOutputSignals.add(anEsterelSignal);
    }
} //for

theInputOutputSignals.trimToSize();
this.myInputOutputSignals =
    (EsterelSignal[]) theInputOutputSignals
    .toArray(new EsterelSignal[theInputOutputSignals.size()]);
theInputSignals.trimToSize();
this.myInputSignals =
    (EsterelSignal[]) theInputSignals
    .toArray(new EsterelSignal[theInputSignals.size()]);
theOutputSignals.trimToSize();
this.myOutputSignals =
    (EsterelSignal[]) theOutputSignals
    .toArray(new EsterelSignal[theOutputSignals.size()]);
theReturnSignals.trimToSize();
this.myReturnSignals = (EsterelSignal[]) theReturnSignals
.toArray(new EsterelSignal[theReturnSignals.size()]);
theSensors.trimToSize();
this.mySensors = (EsterelSignal[]) theSensors
.toArray(new EsterelSignal[theSensors.size()]);
theInputOutputSignals.trimToSize();
this.myInputOutputSignals = (EsterelSignal[]) theInputOutputSignals
.toArray(new EsterelSignal[theInputOutputSignals.size()]);
//get relations
aXPath = this.myXPath
       + "[local-name(.)='Exclusion']";
anElementList = DOMHelpers.getElements(aXPath,
    theparser.getXMLDocument());
ArrayList theRelations = new ArrayList();
for (int i = 0; i < anElementList.size(); i++) {
    theRelations.add(new EsterelRelation(
           theparser,
           this.myXPath
                  + "[local-name(.)='"
                  + ((Element) anElementList.get(i)).getName()
                  + "'][@id='"
                  + ((Element) anElementList.get(i))
                  .getAttributeValue("id")
                  + "']/*[not(local-name(.)='EOV')]"
                  + "[not(local-name(.)='NULL')]"));
} //for
aXPath = this.myXPath
       + "[local-name(.)='Implication']";
anElementList = DOMHelpers.getElements(aXPath,
    theparser.getXMLDocument());
for (int i = 0; i < anElementList.size(); i++) {
    theRelations.add(new EsterelRelation(
           theparser,
           this.myXPath
                  + "[local-name(.)='"
                  + ((Element) anElementList.get(i)).getName()
                  + "'][@id='"
                  + ((Element) anElementList.get(i))
                  .getAttributeValue("id")
                  + "']/*[not(local-name(.)='EOV')]"
```

```
                        + "[not(local-name(.)='NULL')]"));
    } //for
    theRelations.trimToSize();
    this.myRelations = (EsterelRelation[]) theRelations
    .toArray(new EsterelRelation[theRelations.size()]);
    Arrays.sort(this.myRelations,
        new EsterelDeclarationSortByID());
} //public parse

/**
 * Returns a string representation of an esterel module.
 *
 * @return a String representation of an esterel statment
 * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
 */
public final String toString() {
    String aModuleString = "Module "
           + this.getName() + ":\n";
    String aInputString = "";
    String aOutputString = "";
    String aInputOutputString = "";
    if (this.myTyps.length > 0) {
        aModuleString += " type ";
    }
    for (int i = 0; i < this.myTyps.length; i++) {
        aModuleString += this.myTyps[i].toString()
               + " \n";
    }
    if (this.myConstants.length > 0) {
        aModuleString += " constant ";
    }
    for (int i = 0; i < this.myConstants.length; i++) {
        aModuleString += this.myConstants[i].toString(this)
               + " \n";
    }
    if (this.myFunctions.length > 0) {
        aModuleString += " function ";
    }
    for (int i = 0; i < this.myFunctions.length; i++) {
        aModuleString += this.myFunctions[i].toString(this)
               + " \n";
    }
    if (this.myProcedures.length > 0) {
        aModuleString += " procedure ";
    }
    for (int i = 0; i < this.myProcedures.length; i++) {
        aModuleString += this.myProcedures[i].toString(this)
               + " \n";
    }
    if (this.myTasks.length > 0) {
        aModuleString += " task ";
    }
    for (int i = 0; i < this.myTasks.length; i++) {
        aModuleString += this.myTasks[i].toString(this)
               + " \n";
    }
    if (this.myInputSignals.length != 0) {
        for (int i = 0; i < this.myInputSignals.length; i++) {
            aInputString += this.myInputSignals[i].toString(this)
                   + " ";
        }
        aModuleString += "InputSignals ";
        aModuleString += aInputString
               + "\n";
    }
    if (this.myOutputSignals.length != 0) {
        for (int i = 0; i < this.myOutputSignals.length; i++) {
            aOutputString += this.myOutputSignals[i].toString(this)
                   + " ";
        }
        aModuleString += "OutputSignals ";
        aModuleString += aOutputString
               + "\n";
    }
    if (this.myInputOutputSignals.length != 0) {
        for (int i = 0; i < this.myInputOutputSignals.length; i++) {
            aInputOutputString +=
                this.myInputOutputSignals[i].toString(this)
                    + " ";
        }
        aModuleString += "InputoutputSignals ";
        aModuleString += aInputOutputString
               + "\n";
    }
    if (this.mySensors.length > 0) {
        aModuleString += " sensors ";
    }
    for (int i = 0; i < this.mySensors.length; i++) {
        aModuleString += this.mySensors[i].toString(this)
               + " \n";
    }
    if (this.myRelations.length > 0) {
        aModuleString += " relations ";
    }
    for (int i = 0; i < this.myRelations.length; i++) {
        aModuleString += this.myRelations[i].toString(this)
               + " \n";
    }
    aModuleString += ((EsterelStatement) this.anEsterelProgram
           .get(0))
           .toString(this);
    aModuleString += "end module "
           + this.getName();
    return aModuleString;
}
/**
 * @return Returns the aEsterelStateChart.
 */
public final StateChart getEsterelStateChart() {
    return this.anEsterelStateChart;
}
}
```

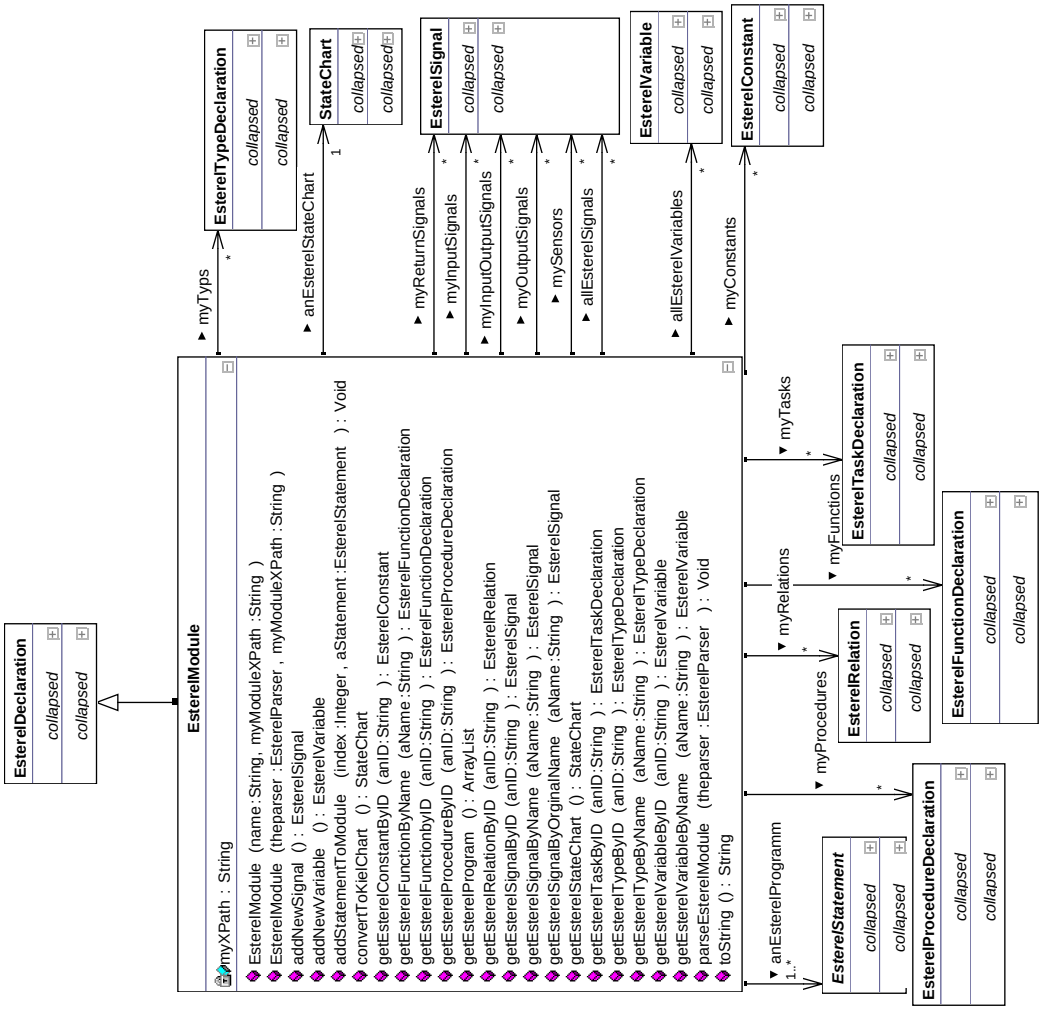


Abbildung C.12.: Klassediagramm `EsterelModule`

C.2.18. EsterelProcedureDeclaration

Der Klassenaufbau ist in der Abbildung C.2.18 dargestellt.

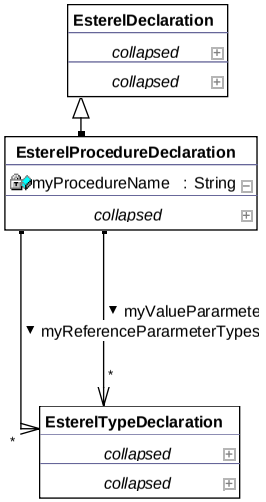


Abbildung C.13.: Klassediagramm `EsterelProcedureDeclaration`

Auflistung C.23: Die Klasse `EsterelProcedureDeclaration`

```
package kiel.fileInterface.esterel.esterel2estudio;
import java.util.ArrayList;
import java.util.List;
//
import kiel.util.DOMHelpers;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
//
import org.jdom.Document;
import org.jdom.Element;
/**
 * <p> It represents esterel procedures .</p>
 *  <p>Copyright: Copyright (c) 2005</p>
 * <p>Company: Uni Kiel</p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
 * @version $Revision: 1.19 $ last modified $Date: 2006/02/06 18:35:29 $
 * <br>
 */
public class EsterelProcedureDeclaration extends EsterelDeclaration {
    /**
     * the procedure name.
     */
    private String myProcedureName;
    /**
     * the reference parameters.
     */
    private ArrayList myReferenceParameterTyps;
    /**
     * the value parameters.
```

```
     */
    private ArrayList myValueParameterTyps;
    /**
     * the simple constructor.
     *
     */
    public EsterelProcedureDeclaration() {
        super();
        this.myReferenceParameterTyps = null;
        this.myValueParameterTyps = null;
        this.myProcedureName = "";
    }
    /**
     * Creates an EsterelProcedure and fills the class variables.
     * <br>Sideeffects: class variables are changed.
     * @param theparser the esterel parser
     * @param theXPath the XPath to the EsterelVariable
     * @throws EsterelParserException if something
     * goes wrong at the parsing.
     */
    public EsterelProcedureDeclaration(
        final EsterelParser theparser,
        final String theXPath)
        throws EsterelParserException {
        super();
        this.myReferenceParameterTyps = null;
        this.myValueParameterTyps = null;
        this.myProcedureName = "";
        if (theXPath != null && theparser != null) {
            this.parseEsterelStatement(theparser, theXPath);
        } else {
            throw new EsterelParserException(
"ProcedureDelclaration : no parser or path");
        }
    }
    /**
       * Retuns the type of the DelayExpession.
       * <br> Sideeffects: none
       * @param aDelayList the cildren elements of the DelayExpression
       * @param anElementIndex the number of the DelayElement in aDelayList
       * @param xmldoc the <code>org.jdom.Document</code>
       * reprsentation of the .exp file
       * @return the type name
       * @throws EsterelParserException if no type is found.
       */
    private String getType(
        final List aDelayList,
        final int anElementIndex,
        final Document xmldoc)
    throws EsterelParserException {
        if (((Element) aDelayList.get(anElementIndex)).getName() == "Ref") {
            String aRefXPathSearchString =
                "//*[@id='"
                    + (
                        (Element) aDelayList.get(
                            anElementIndex)).getAttributeValue(
                                "id")
                    + "']/*[not(local-name(.)='EOV')]"
                    + "[not(local-name(.)='NULL')]";
            List theRefElementList =
                DOMHelpers.getElements(aRefXPathSearchString, xmldoc);
            if (theRefElementList.size() > 0) {
                return ((Element) theRefElementList.get(0)).getText();
            }
            }
        throw new EsterelParserException("EsterelProceddure: foun no type");
    }
    /**
      * Parses a procedure.
      * <br> Sideeffects: sets the class variables.
      * @param theXPath the XPath which leads to the Expression
      * @param theparser the esterel parser
      * @throws EsterelParserException if there is a parsing problem
      * @see kiel.fileInterface.esterel.EsterelParserException
      */
    public final void parseEsterelStatement(
        final EsterelParser theparser,
        final String theXPath)
        throws EsterelParserException {
        List anElementList =
            DOMHelpers.getElements(theXPath, theparser.getXMLDocument());
        if (anElementList.size() > 0) {
            this.setName(((Element) anElementList.get(0))
                .getParentElement().getName());
            this.setXMLID(((Element) anElementList.get(0))
                .getParentElement().getAttributeValue("id"));
        }
        int theEOVCounter = 0;
        int i = 0;
        this.myReferenceParameterTyps = new ArrayList();
        this.myValueParameterTyps = new ArrayList();
        while (theEOVCounter < 2 && i < anElementList.size()) {
            Element anElement = (Element) anElementList.get(i);
            if (anElement.getName().compareTo("EOV") == 0) {
                theEOVCounter++;
            } else if (i == 0) { // parse name}
                this.myProcedureName = anElement.getText();
            } else if (theEOVCounter == 0) {
                this.myReferenceParameterTyps.add(
                    this.getType(anElementList, i, theparser.getXMLDocument()));
            } else if (theEOVCounter == 1) {
                this.myValueParameterTyps.add(
                    this.getType(anElementList, i, theparser.getXMLDocument()));
            }
            i++;
        } //while
    }
    /**
     * Returns the String representation of the variable.
     * @param anEsterelModule an <code>EsterelModule</code>.
     * @return the String representation of an EsterelVariable
     */
```

```
public final String toString(final EsterelModule anEsterelModule) {
    String result = this.myProcedureName + " ( ";
    if (this.myReferenceParameterTyps != null) {
        for (int i = 0; i < this.myReferenceParameterTyps.size(); i++) {
            result += (String) this.myReferenceParameterTyps.get(i) + " ,";
        }
    }
    result += " ) ( ";
    if (this.myValueParameterTyps != null) {
        for (int i = 0; i < this.myValueParameterTyps.size(); i++) {
            result += (String) this.myValueParameterTyps.get(i) + " ,";
        }
        result += " )";
        return result;
    }
    /**
     * @return Returns the myProcedureName.
     */
    public final String getProcedureName() {
        return this.myProcedureName;
    }
}
```

C.2.19. EsterelRelation

Der Klassenaufbau ist in der Abbildung C.2.19 dargestellt.

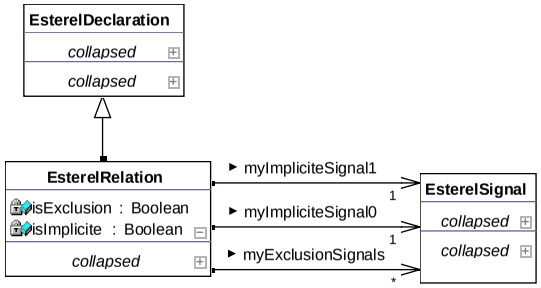


Abbildung C.14.: Klassediagramm `EsterelRelation`

Auflistung C.24: Die Klasse `EsterelRelation`

```
package kiel.fileInterface.esterel.esterel2estudio;

import java.util.ArrayList;
import java.util.List;
//
import kiel.util.DOMHelpers;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
//
import org.jdom.Document;
import org.jdom.Element;

/**
 * <p>
 *  It represents esterel Relations .
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.19 $ last modified $Date: 2006/02/06 18:35:29 $
 *             <br>
 */
public class EsterelRelation extends EsterelDeclaration {
    /**
     * true if is explicite.
     */
    private boolean isExclusion;

    /**
     * true if is implicite.
     */
    private boolean isImplicite;

    /**
     * the explicite signals.
     */
    private ArrayList myExclusionSignals;

    /**
     * the implite signal 0.
     */
    private String myImpliciteSignal0;

    /**
     * the implite signal 1.
     */
    private String myImpliciteSignal1;
```

```
    /**
     * the simple constructor.
     */
    public EsterelRelation() {
        super();
        this.isExclusion = false;
        this.isImplicite = false;
        this.myImpliciteSignal0 = "";
        this.myImpliciteSignal1 = "";
        this.myExclusionSignals = null;
    }

    /**
     * Creates an EsterelRelation and fills the class variables. <br>
     * Sideeffects: class variables are changed.
     *
     * @param theparser
     *                the esterel parser
     * @param theXPath
     *                the XPath to the EsterelVariable
     * @throws EsterelParserException
     *                if something goes wrong at the parsing.
     */
    public EsterelRelation(final EsterelParser theparser, final String theXPath)
            throws EsterelParserException {
        super();
        this.isExclusion = false;
        this.isImplicite = false;
        this.myImpliciteSignal0 = null;
        this.myImpliciteSignal1 = null;
        this.myExclusionSignals = null;
        if (theXPath != null && theparser != null) {
            this.parseEsterelStatement(theparser, theXPath);
        } else {
            throw new EsterelParserException("RelationDelclaration "
                    + ": no parser or path");
        }
    }

    /**
     * Retuns the signal of the Signalexpession. <br>
     * Sideeffects: none
     *
     * @param anElement
     *                an xml element
     * @param xmldoc
     *                the <code>org.jdom.Document</code> reprsentation of the
     *                .exp file
     * @return the signals name
     * @throws EsterelParserException
     *                if no signal is found.
     */
    private String getSignal(final Element anElement, final Document xmldoc)
            throws EsterelParserException {
        if (anElement.getName().compareTo("Ref") == 0) {
            String aRefXPathSearchString = "//*[@id='"
                    + anElement.getAttributeValue("id")
                    + "']/*[not(local-name(.)='EOV')]"
                    + "[not(local-name(.)='NULL')]";
            List theRefElementList = DOMHelpers.getElements(
                    aRefXPathSearchString,
                    xmldoc);
            if (theRefElementList.size() > 0
                    && ((Element) theRefElementList.get(1)).getParentElement()
                            .getName() == "SignalSymbol") {
                return ((Element) theRefElementList.get(0)).getText();
            }
        }
        throw new EsterelParserException("EsterelRelation: found no signal");
    }

    /**
     * Parses a Relation. <br>
     * Sideeffects: sets the class variables.
     *
     * @param theXPath
     *                the XPath which leads to the Expression
     * @param theparser
     *                the esterel parser
     * @throws EsterelParserException
     *                if there is a parsing problem
     * @see kiel.fileInterface.esterel.EsterelParserException
     */
    public final void parseEsterelStatement(
            final EsterelParser theparser,
            final String theXPath) throws EsterelParserException {
        List anElementList = DOMHelpers.getElements(theXPath, theparser
                .getXMLDocument());
        Element anElement = (Element) anElementList.get(0);
        if (anElement.getParentElement().getName().compareTo("Implication")
                == 0) {
            this.myImpliciteSignal0 = this.getSignal(anElement, theparser
                    .getXMLDocument());
            anElement = (Element) anElementList.get(1);
            this.myImpliciteSignal1 = this.getSignal(anElement, theparser
                    .getXMLDocument());
            this.isImplicite = true;
        } else if (anElement.getParentElement().getName()
                .compareTo("Exclusion") == 0) {
            this.isExclusion = true;
            this.myExclusionSignals = new ArrayList();
            for (int i = 0; i < anElementList.size(); i++) {
                anElement = (Element) anElementList.get(i);
                this.myExclusionSignals.add(this.getSignal(anElement, theparser
                        .getXMLDocument()));
            }
        }
    }

    /**
     * Returns the String representation of the variable.
```

```
    for (int i = 0; i < this.myExclusionSignals.size() - 1; i++) {
      result += (String) this.myExclusionSignals.get(0) + " # ";
    }
    result += (String) this.myExclusionSignals
        .get(this.myExclusionSignals.size() - 1);
  }
  return result;
}
}
```

```
 *@param anEsterelModule an <code>EsterelModule</code>
 * @return the String representation of an EsterelVariable
 * @see EsterelModule
 */
public final String toString(final EsterelModule anEsterelModule) {
    String result = "";
    if (this.isImplicite) {
      result = this.myImpliciteSignal0 + " => " + this.myImpliciteSignal1;
    } else if (this.isExclusion) {
```

C.2.20. EsterelSignalExpression

Der Klassenaufbau ist in der Abbildung C.2.20 dargestellt.

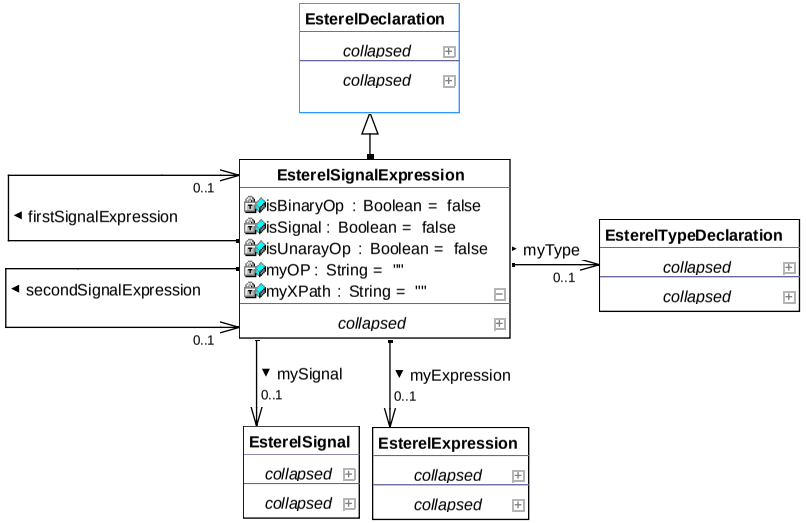


Abbildung C.15.: Klassediagramm `EsterelSignalExpression`

Auflistung C.25: Die Klasse `EsterelSignalExpression`

```
package kiel.fileInterface.esterel.esterel2estudio;

import java.util.ArrayList;
import java.util.List;
import kiel.dataStructure.CompoundLabel;
import kiel.dataStructure.StringLabel;
import kiel.dataStructure.TransitionLabel;
import kiel.dataStructure.boolexp.BooleanExpression;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.CompoundLabelException;
import kiel.util.CompoundLabelParser;
import kiel.util.DOMHelpers;
import org.jdom.Document;
import org.jdom.Element;

/**
 * <p>
 * Represents a SignalExpression in esterel.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2004
 * </p>
 * <p>
 * Company: Uni Kiel
```

```
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.43 $ last modified $Date: 2006/02/06 18:35:29 $
 */
public class EsterelSignalExpression extends EsterelDeclaration {
   /**
    * If the SignalExpression has one or more sub SignalExpression then
    * the first is put into this member.
    */
   private EsterelSignalExpression firstSignalExpression;

   /**
    * <code>true</code> if the SignalExpression is a binary operation
    * else <code>false</code>.
    */
   private boolean isBinaryOp;

   /**
    * <code>true</code>, if the SignalExpression is only a Signal else
    * <code>false</code>.
    */
   private boolean isSignal;

   /**
    * <code>true</code>, if the SignalExpression isunary operation else
    * <code>false</code>.
    */
   private boolean isUnarayOp;

   /**
    * the operation name.
    */
   private String myOP;

   /**
    * the signal name.
    */
   private EsterelSignal mySignal;

   /**
    * the SignalExpressions type name.
    */
   private EsterelTypeDeclaration myType;

   /**
    * the XPath to the SignalExpression in an .exp file.
    */
   private String myXPath;

   /**
    * if the SignalExpression has two sub SignalExpression then the second
    * is put into this member.
    */
   private EsterelSignalExpression secondSignalExpression;

   /**
    * The standard constructor.
    */
   public EsterelSignalExpression() {
      super();
      this.myOP = "";
      this.mySignal = null;
      this.isBinaryOp = false;
      this.isSignal = false;
      this.isUnarayOp = false;
      this.firstSignalExpression = null;
      this.secondSignalExpression = null;
      this.myXPath = "";
   } //public EsterelSignalExpression() {

   /**
    * Uses the standard constructor and call the parseSignalExpression
    * method. <br>
    * Sideeffects: sets the class variables
    *@param theparser an esterel parser.
    * @param aXPath
    *                the XPath which leads to this Signalexpression in the
    *                <br>
    *                <code>org.jdom.Document</code> respresentation of an
    *                .exp file
    * @param xmldoc
    *                the <code>org.jdom.Document</code>
    * @throws EsterelParserException
    *                 is only delivered
    * @see kiel.fileInterface.esterel.EsterelParserException
    */
   public EsterelSignalExpression(
          final EsterelParser theparser,
          final String aXPath,
          final Document xmldoc)
          throws EsterelParserException {
      this();
      this.parseSignalExpression(
          theparser,
              aXPath,
              xmldoc);
   }

   /**
    * Converts the signalexpression to a
    * <code>kiel.dataStructure.TransitionLabel</code>.
    *
    * @param anEsterelModule
    *                the esterel module
    * @param theLocalEvents
    *                all local events
    * @param theLocalVariables
    *                all local variables
    * @param aTransLabel
    *                an old label or null
    * @return a <code>kiel.dataStructure.TransitionLabel</code>
```

```
     * @see kiel.dataStructure.TransitionLabel
     */
    public final TransitionLabel convertToKiel(
            final EsterelModule anEsterelModule,
            final ArrayList theLocalEvents,
            final ArrayList theLocalVariables,
            final TransitionLabel aTransLabel) {
        CompoundLabelParser.getInstance();
        // Setting StateChart for identifier searching
        CompoundLabelParser.setStateChart(anEsterelModule
                .getEsterelStateChart());
        CompoundLabelParser.setAllLocals(
                theLocalVariables,
                theLocalEvents);
        String aStringLabel = this.toString(anEsterelModule).trim();
        TransitionLabel aLabel = null;
        BooleanExpression aBoolExp = null;
        try {
            aBoolExp = CompoundLabelParser
                    .parseBooleanExpression(aStringLabel);
        } catch (CompoundLabelException ex) {
            return aLabel = new StringLabel(
                    aStringLabel);
        }
        if (aTransLabel != null) {
            if (aTransLabel instanceof StringLabel) {
                aLabel = new StringLabel(
                        aStringLabel);
            } else {
                aLabel = aTransLabel;
                ((CompoundLabel) aLabel).getTrigger().setEventExpression(
                        aBoolExp);
            }
        } else {
            aLabel = new CompoundLabel();
            ((CompoundLabel) aLabel).getTrigger().setEventExpression(
                    aBoolExp);
        }
        return aLabel;
    } // covert

    /**
     * Retuns the operation of the Signalexpession. <br>
     * Sideeffects: none
     *
     * @param anElementList
     *                the cildren elements of the SignalExpression
     * @return the operations nam
     * @throws EsterelParserException
     *                 if no operation is found.
     */
    private String getOperation(
            final List anElementList)
            throws EsterelParserException {
        if (((Element) anElementList.get(1)).getName() == "S") {
            return ((Element) anElementList.get(1)).getText();
        }
        throw new EsterelParserException(
                "EsterelSignalExpression: found no operation");
    }

    /*
     * Retuns the signal of the Signalexpession. <br>
     * Sideeffects: none
     *
     * @param anElementList
     *                the cildren elements of the SignalExpression
     * @param xmldoc
     *                the <code>org.jdom.Document</code> reprsentation of the
     *                .exp file
     * @return the signals name
     * @throws EsterelParserException
     *                 if no signal was found
     *//*
    private String getSignal(
            final List anElementList,
            final Document xmldoc)
            throws EsterelParserException {
        final int theSignalIndex = 1;
        if (((Element) anElementList.get(theSignalIndex)).getName() == "Ref") {
            String aRefXPathSearchString = "//*[@id='"
                    + ((Element) anElementList.get(theSignalIndex))
                            .getAttributeValue("id")
                    + "']/*[not(local-name(.)='EOV')]"
                    + "[not(local-name(.)='NULL')]";
            List theRefElementList = DOMHelpers.getElements(
                    aRefXPathSearchString,
                    xmldoc);
            if (theRefElementList.size() > 0
                    && (((Element) theRefElementList.get(1))
                            .getParentElement()
                               .getName()
                               .compareTo(
                                       "SignalSymbol") == 0
                                || ((Element) theRefElementList
                            .get(1)).getParentElement().getName().compareTo(
                            "BuiltinSignalSymbol") == 0)) {
                return ((Element) theRefElementList.get(0)).getText();
            }
            throw new EsterelParserException(
                    "EsterelSignalExpression: found no signal");
        } else if (((Element) anElementList.get(theSignalIndex))
                .getName() == "SignalSymbol") {
            return ((Element) anElementList.get(theSignalIndex)).getChild(
                    "S").getText();
        }
        throw new EsterelParserException(
                "EsterelSignalExpression: found no signal");
    }
*/
    /**
     * Retuns an <code>EsterelSignalExpression</code> of a sub
```

```
     * Signalexpession. <br>
     * Sideeffects: none
     *
     * @param anElementList
     *                the cildren elements of the SignalExpression
     * @param theparser an<code>EsterelParser</code>
     * @param xmldoc
     *                the <code>org.jdom.Document</code> reprsentation of the
     *                .exp file
     * @throws EsterelParserException
     *                is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     * @return a sub SignalExpression
     */
    private EsterelSignalExpression getSignalExpression(
            final EsterelParser theparser,
            final List anElementList,
            final Document xmldoc)
            throws EsterelParserException {
        final int firstindex = 2;
        final int secondindex = 3;
        int actualElementNumber = 0;
        if (this.firstSignalExpression == null) {
            actualElementNumber = firstindex;
        } else if (this.secondSignalExpression == null) {
            actualElementNumber = secondindex;
        } else {
            actualElementNumber = -1;
        }
        if (actualElementNumber == firstindex) {
            return new EsterelSignalExpression(
                theparser,
                    this.myXPath
                           + "[local-name(.)='"
                           + ((Element) anElementList
                                   .get(actualElementNumber)).getName()
                           + "'][@id='"
                           + ((Element) anElementList
                                   .get(actualElementNumber))
                                   .getAttributeValue("id") + "']/*",
                xmldoc);
        } else if (actualElementNumber == secondindex) {
            return new EsterelSignalExpression(
                theparser,
                    this.myXPath
                           + "[local-name(.)='"
                           + ((Element) anElementList
                                   .get(actualElementNumber)).getName()
                           + "'][@id='"
                           + ((Element) anElementList
                                   .get(actualElementNumber))
                                   .getAttributeValue("id") + "']/*",
                xmldoc);
        } else {
            return null;
        }
    }

    /**
     * Parses a SignalExpression. <br>
     * Sideeffects: sets the class variables
     *@param theparser an esterel parser.
     * @param searchXPath
     *                the XPath which leads to the Signalexpression
     * @param xmldoc
     *                the <code>org.jdom.Document</code> representation of
     *                the .exp file
     * @throws EsterelParserException
     *                is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     */
    public final void parseSignalExpression(
            final EsterelParser theparser,
            final String searchXPath,
            final Document xmldoc)
            throws EsterelParserException {
        int i = 0;
        this.myXPath = searchXPath;
        List anElementList = DOMHelpers
                .getElements(
                        this.myXPath
                   + "[not(local-name(.)='EOV')][not(local-name(.)='NULL')]",
                        xmldoc);
        // the XPath finds the children of the SignalExpression
        String actualElement = ((Element) anElementList.get(i))
                .getParentElement()
                    .getName();
        if (actualElement == "UnaryOp") {
            this.isUnarayOp = true;
            this.myType =
                theparser
                .getEsterelModule(theparser.getactualEsterModule())
                .getEsterelTypeByID(((Element) anElementList.get(0))
                           .getAttributeValue("id"));
            this.myOP = this.getOperation(anElementList);
            this.firstSignalExpression = this.getSignalExpression(
                theparser,
                   anElementList,
                   xmldoc);
        } else if (actualElement == "BinaryOp") {
            this.isBinaryOp = true;
            this.myType =
                theparser
                .getEsterelModule(theparser.getactualEsterModule())
                .getEsterelTypeByID(((Element) anElementList.get(0))
                           .getAttributeValue("id"));
            this.myOP = this.getOperation(anElementList);
            this.firstSignalExpression = this.getSignalExpression(
                theparser,
                anElementList,
                    xmldoc);
            this.secondSignalExpression = this.getSignalExpression(
```

```
        theparser,
            anElementList,
            xmldoc);
    } else if (actualElement == "LoadSignalExpression") {
        this.isSignal = true;
        this.myType =
            theparser
            .getEsterelModule(theparser.getactualEsterModule())
            .getEsterelTypeByID(((Element) anElementList.get(0))
                    .getAttributeValue("id"));
        this.mySignal = theparser
        .getEsterelModule(theparser.getactualEsterModule())
        .getEsterelSignalByID(((Element) anElementList.get(1))
            .getAttributeValue("id"));
    } else {
        throw new EsterelParserException(
            "SignalExpression: cannot identify expression");
    }
}
/**
 * Coverts the SignalExpression to String. <br>
 * Sideeffects: none
 *@param anEsterelModule an<code>EsterelModule</code>
 * @return a string representation of the SignalExpression
 */
public final String toString(final EsterelModule anEsterelModule) {
    String result;
    if (this.isBinaryOp) {
        result = " ( "
            + this.firstSignalExpression
                .toString(anEsterelModule)
            + " "
            + this.myOP
            + " "
            + this.secondSignalExpression
            .toString(anEsterelModule)
            + " ) ";
    } else if (this.isSignal) {
        result = this.mySignal.getSignalIdentifier();
    } else if (this.isUnaryOp) {
        result = this.myOP
                + " ( "
                + this.firstSignalExpression.toString(anEsterelModule)
                + " ) ";
    } else {
        result = "null";
    }
    return result;
}
/**
 * @return true if it is a signal.
 */
public final boolean isSignal() {
    return this.isSignal;
}
/**
 * @return Returns the type..
 */
public final EsterelTypeDeclaration getType() {
    return this.myType;
}
/**
 * @return Returns the mySignal.
 */
public final EsterelSignal getMySignal() {
    return this.mySignal;
}
/**
 * @param aSignal The mySignal to set.
 */
public final void setMySignal(
        final  EsterelSignal aSignal) {
    this.mySignal = aSignal;
}
}
```

C.2.21. EsterelSignal

Der Klassenaufbau ist in der Abbildung C.2.21 dargestellt.

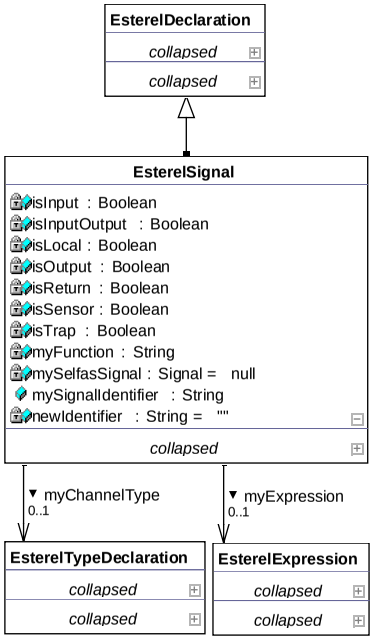


Abbildung C.16.: Klassediagramm `EsterelSignal`

Auflistung C.26: Die Klasse `EsterelSignal`

```
package kiel.fileInterface.esterel.esterel2estudio;

//
import java.util.List;
import java.util.Random;
import kiel.dataStructure.eventexp.CombineWithAdd;
import kiel.dataStructure.eventexp.CombineWithMult;
import kiel.dataStructure.eventexp.IntegerSignal;
import kiel.dataStructure.eventexp.Signal;
import kiel.dataStructure.eventexp.StringSignal;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import org.jdom.Element;
```

```
/**
 * <p>
 *  It represents esterel signals as global, local trap and
 * extern signals.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.40 $ last modified $Date: 2006/02/13 14:00:04 $
 *
 */
public class EsterelSignal
       extends EsterelDeclaration {
    /**
     * Is true if it is an input signal.
     */
    private boolean isInput; // child 3

    /**
     * Is true if it is an input and output signal.
     */
    private boolean isInputOutput;

    /**
     * Is true if it is a local signal.
     */
    private boolean isLocal;

    /**
     * Is true if it is an output signal.
     */
    private boolean isOutput;

    /**
     * Is true if it is an return signal.
     */
    private boolean isReturn;

    /**
     * Is true , if it is a sensor signal.
     */
    private boolean isSensor;

    /**
     * Is true if it is trap signal.
     */
    private boolean isTrap;

    /**
     * The type of the signal.(Optional).
     */
    private EsterelTypeDeclaration myChannelType; // child1

    /**
     * The expression of the signal.(Optional).
     */
    private EsterelExpression myExpression; //child

    // 2
    /**
     * The function of the signal.(Optional).
     */
    private EsterelFunctionDeclaration myFunction = null;

    /**
     * The EsterelSignal as signal.
     */
    private Signal mySelfasSignal = null;

    /**
     * The name of the signal.
     */
    private String mySignalIdentifier; // child 0

    /**
     * The new name.
     */
    private String newIdentifier = "";

    /**
     * @param anEsterelSignal
     *                a esterelSignal.
     * @return an integer.
     */
    /**
     * the standart constructor.
     */
    public EsterelSignal() {
        super();
    }

    /**
     * @param theparser
     *                a <code>EsterelParser</code>.
     * @param anElement
     *                an <code>Element</code>.
     * @throws EsterelParserException
     *                the exception.
     */
    public EsterelSignal(final EsterelParser theparser,
           final Element anElement)
           throws EsterelParserException {
        this(
               theparser,
               "//*[@id='" + anElement.getAttributeValue("id")
                       + "']/*");
        this.setName(anElement.getName());
```

```
        this.setXMLID(anElement.getAttributeValue("id"));
    }

    /**
     * Creates an EsterelSignal an fills the class variables. <br>
     * Sideeffects: class variables are changed.
     *
     * @param theparser
     *                the esterel parser
     * @param theXPath
     *                the XPath to the EsterelSignal
     * @throws EsterelParserException
     *                if something goes wrong at the parsing.
     */
    private EsterelSignal(final EsterelParser theparser,
            final String theXPath)
            throws EsterelParserException {
        this.isSensor = false;
        this.isTrap = false;
        this.isLocal = false;
        this.myFunction = null;
        this.isInput = false;
        this.isOutput = false;
        this.isInputOutput = false;
        this.isReturn = false;
        this.mySignalIdentifier = null;
        this.myExpression = null;
        this.myChannelType = null;
        if (theparser != null) {
            parseEsterelStatement(theparser,
                theXPath);
        }
    }

    /**
     * Coverts it to the KIEL datastructure Signal.
     *
     * @param anEsterelModule
     *                the module
     * @return a Signal
     * @throws Esterel2EstudioException
     *                if it cannot converted
     */
    public final Signal convertToKiel(
            final EsterelModule anEsterelModule)
            throws Esterel2EstudioException {
        Signal result = this.mySelfasSignal;
        if (result == null) {
            this.createNewIdentifierName(anEsterelModule);
            if (this.myFunction != null && this.myChannelType != null) {
                if (this.myFunction.getFunctionIdentifier().compareTo("+") == 0) {
                    if (this.myExpression != null) {
                        result = new CombineWithAdd(
                                this.getSignalIdentifier(),
                                this.myExpression
                                .convertToKiel(anEsterelModule));
                    } else {
                        result = new CombineWithAdd(
                                this.getSignalIdentifier());
                    }
                } else if (this.myFunction.getFunctionIdentifier()
                        .compareTo("*") == 0) {
                    if (this.myExpression != null) {
                        result = new CombineWithMult(
                                this.getSignalIdentifier(),
                                this.myExpression
                                .convertToKiel(anEsterelModule));
                    } else {
                        result = new CombineWithMult(
                                this.getSignalIdentifier());
                    }
                } else {
                    throw new Esterel2EstudioException(
                            "function does not exist in datastructure "
                            + this.myFunction);
                }
            } else if (this.myChannelType != null) {
                if (this.myChannelType.
                        getTypeName()
                        .compareTo("boolean") == 0) {
                    result = new Signal(
                            this.getSignalIdentifier());
                } else if (this.myChannelType
                        .getTypeName().compareTo("string") == 0) {
                    if (this.myExpression != null) {
                        result = new StringSignal(
                                this.getSignalIdentifier(),
                                this.myExpression
                                .convertToKielString(anEsterelModule));
                    } else {
                        result = new StringSignal(
                                this.getSignalIdentifier());
                    }
                } else if (this.myChannelType
                        .getTypeName().compareTo("integer") == 0) {
                    if (this.myExpression != null) {
                        result = new IntegerSignal(
                                this.getSignalIdentifier(),
                                this.myExpression
                                .convertToKiel(anEsterelModule));
                    } else {
                        result = new IntegerSignal(
                                this.getSignalIdentifier());
                    }
                } else if (this.myChannelType
                        .getTypeName().compareTo("float") == 0) {
                    if (this.myExpression != null) {
                        result = new IntegerSignal(
                                this.getSignalIdentifier(),
                                this.myExpression
                                .convertToKiel(anEsterelModule));
                    } else {
```

```
                    result = new IntegerSignal(
                            this.getSignalIdentifier());
                }
            } else if (this.myChannelType
                    .getTypeName().compareTo("double") == 0) {
                if (this.myExpression != null) {
                    result = new IntegerSignal(
                            this.getSignalIdentifier(),
                            this.myExpression
                            .convertToKiel(anEsterelModule));
                } else {
                    result = new IntegerSignal(
                            this.getSignalIdentifier());
                }
            } else {
                if (this.myExpression != null) {
                    result = new StringSignal(
                            this.getSignalIdentifier(),
                            this.myExpression
                            .convertToKielString(anEsterelModule));
                } else {
                    result = new StringSignal(
                            this.getSignalIdentifier());
                }
                /*
                throw new Esterel2EstudioException(
                        "EsterelSignal :" + "Unknown Type :"
                                + this.myChannelType.toString());
                                */
            }
        } else {
            result = new Signal(
                    this.getSignalIdentifier());
        }
        this.mySelfasSignal = result;
    }
    return result;
}

/**
 * Creates a new name for a trap signal, the new name is the preName
 * form the preferences the name the postName of the preferences and
 * eventually a random number.
 *
 * @param anEsterelModule
 *                the esterel Module
 */
public final void createNewIdentifierName(
        final EsterelModule anEsterelModule) {
    if (this.isTrap) {
        String preName = Esterel2EstudioProperties.getPreSting();
        String postName = Esterel2EstudioProperties.getPostString();
        if (preName == null) {
            preName = "";
        }
        if (postName == null) {
            postName = "";
        }
        Random rand = new Random();
        this.newIdentifier = preName + this.mySignalIdentifier
                + postName;
        while (anEsterelModule
                .getEsterelSignalByOrginalName(this.newIdentifier)
                != null) {
            this.newIdentifier = preName + this.mySignalIdentifier
                    + postName
                    + rand.nextInt();
        }
    }
}

/**
 * @return the type of the signal
 */
public final EsterelTypeDeclaration getChannelType() {
    return this.myChannelType;
}

/**
 * @return the expression of the signal
 */
public final EsterelExpression getExpression() {
    return this.myExpression;
}

/**
 * @return the name of the function
 */
public final String getFunction() {
    return this.myFunction.getFunctionIdentifier();
}

/**
 * USE CAREFULLY.
 *
 * @return the orginalSignalIdentifier.
 */
public final String getOrginalSignalIdentifier() {
    return this.mySignalIdentifier;
}

/**
 * @return the signal identifier.
 */
public final String getSignalIdentifier() {
    if (this.newIdentifier == "") {
        return this.mySignalIdentifier;
    }
    return this.newIdentifier;
}

/**
```

```
     * @return is <code>true</code> if signal is an input signal.
     */
    public final boolean isInput() {
        return this.isInput;
    }

    /**
     * @return is <code>true</code> if signal is an input and output
     *             signal.
     */
    public final boolean isInputOutput() {
        return this.isInputOutput;
    }

    /**
     * @return is <code>true</code> if signal is local signal.
     */
    public final boolean isLocal() {
        return this.isLocal;
    }

    /**
     * @return is <code>true</code> if signal is an output signal.
     */
    public final boolean isOutput() {
        return this.isOutput;
    }

    /**
     * @return is <code>true</code> if signal is an return signal.
     */
    public final boolean isReturn() {
        return this.isReturn;
    }

    /**
     * @return is <code>true</code> if signal is an sensor signal.
     */
    public final boolean isSensor() {
        return this.isSensor;
    }

    /**
     * @return Returns the isTrap.
     */
    public final boolean isTrap() {
        return this.isTrap;
    }

    /**
     * Parses a signal. <br>
     * Sideeffects: sets the class variables.
     *
     * @param theXPath
     *                the XPath which leads to the Expression
     * @param theparser
     *                the esterel parser
     * @throws EsterelParserException
     *                if there is a parsing problem
     * @see kiel.fileInterface.esterel.EsterelParserException
     */
    public final void parseEsterelStatement(
            final EsterelParser theparser,
            final String theXPath)
            throws EsterelParserException {
        final int aTHREE = 3;
        final int aFOUR = 4;
        List anElementList = DOMHelpers.getElements(theXPath,
                theparser.getXMLDocument());
        if (anElementList.size() > 0) {
        this.setName(((Element) anElementList.get(0))
                .getParentElement().getName());
        this.setXMLID(((Element) anElementList.get(0))
                .getParentElement().getAttributeValue("id"));
        for (int i = 0; i < anElementList.size(); i++) {
            if (i == 0
                    && ((Element) anElementList.get(i)).getName() != "NULL") {
                this.mySignalIdentifier = new String(
                        ((Element) anElementList.get(i)).getText());
            } else if (i == 1
                    && ((Element) anElementList.get(i)).getName() != "NULL") {
                this.myChannelType =
                    theparser
                    .getEsterelModule(theparser.getactualEsterModule())
                    .getEsterelTypeByID(
                        ((Element) anElementList.get(i))
                        .getAttributeValue("id"));
            } else if (i == 2
                    && ((Element) anElementList.get(i)).getName() != "NULL") {
                this.myExpression = new EsterelExpression(
                    theparser,
                        theXPath + "[local-name(.)='"
                                + ((Element) anElementList.get(i)).getName()
                                + "'][@id='"
                                + ((Element) anElementList.get(i))
                                .getAttributeValue("id")
                                + "']/*",
                        theparser.getXMLDocument());
            } else if (i == aTHREE
                    && ((Element) anElementList.get(i)).getName() != "NULL") {
                String aSignalType = ((Element) anElementList.get(i)).getText();
                if (aSignalType.equals("0")) {
                    this.isInput = true;
                } else if (aSignalType.equals("1")) {
                    this.isOutput = true;
                } else if (aSignalType.equals("2")) {
                    this.isInputOutput = true;
                } else if (aSignalType.equals("3")) {
                    this.isSensor = true;
                } else if (aSignalType.equals("4")) {
                    this.isReturn = true;
                } else if (aSignalType.equals("5")) {
```

```
      this.isLocal = true;
    } else if (aSignalType.equals("6")) {
      this.isTrap = true;
    } else {
      throw new EsterelParserException(
          "Unknow Signal Type: " + aSignalType);
    }
  } else if (i == aFOUR
      && ((Element) anElementList.get(i)).getName() != "NULL") {
    this.myFunction =
      theparser
        .getEsterelModule(theparser.getactualEsterModule())
        .getEsterelFunctionbyID(((Element) anElementList.get(i))
            .getAttributeValue("id"));
      /*new EsterelFunction(theparser,
          "//*[@id='"
          + ((Element) anElementList.get(i))
          .getAttributeValue("id")
          + "']/*",
          theparser.getXMLDocument());
          */
  } else if (i == 0) {
      throw new EsterelParserException(
          "No Signalname");
    }
  }
  } else {
    throw new EsterelParserException(
"No SignalDeclaration found");
    }
}
/**
 * @param aSignalIdentifier
 *            The mySignalIdentifier to set.
 */
public final void setMySignalIdentifier(
    final String aSignalIdentifier) {
  this.mySignalIdentifier = aSignalIdentifier;
}
/**
 * @param setTrap
 *            The isTrap to set.
 */
public final void setTrap(
    final boolean setTrap) {
  this.isTrap = setTrap;
}

/**
 * Creats a string representation of the <code>EsterelSignal</code>.
 *
 * @param anEsterelModule
 *            a <code>EsterelModule</code>
 * @return <code>this.toString()</code>
 * @see EsterelModule
 */
public final String toString(
    final EsterelModule anEsterelModule) {
  String result = this.getSignalIdentifier();
  if (this.myExpression != null) {
    result += " := " + this.myExpression.toString(anEsterelModule);
  }
  if (this.myFunction != null) {
    result += " combine ";
  }
  if (this.myChannelType != null) {
    result += " : " + this.myChannelType;
  }
  if (this.myFunction != null) {
    result += " with " + this.myFunction;
  }
  return result + "\n";
}
}
```

C.2.22. EsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.22 dargestellt.

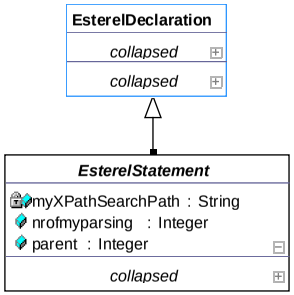


Abbildung C.17.: Klassediagramm `EsterelStatement`

Auflistung C.27: Die Klasse `EsterelStatement`

```
package kiel.fileInterface.esterel.esterel2estudio;

//
import java.util.ArrayList;

import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;

/**
 * <p>
 *  This is the abstract superclass of all esterel statements
 * classes.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2004
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.44 $ last modified $Date: 2006/02/06 18:35:29 $
 */
public abstract class EsterelStatement
       extends EsterelDeclaration {

/** a XPath search string which leads to this statement. */
private String myXPathSearchPath;
 /**
 * Simple constructor.
 */
public EsterelStatement() {
    this(
            null);
} //public EsterelStatement()

/**
 * Parses a Document and set the class varaibles. <br>
 * It uses the <code>preParseEsterelStatement</code> methode.
 *
 * @param theparser
 *                  an intace of a EsterelParser
 * @see #preParseEsterelStatement(EsterelParser)
 * @see kiel.fileInterface.esterel.EsterelParser
 */
public EsterelStatement(final EsterelParser theparser) {
    this.myXPathSearchPath = "";
    if (theparser != null) {
        this.preParseEsterelStatement(theparser);
    }
} //public EsterelStatement( EsterelParser theparser)

/**
```

```
     * Returns a state representation of the esterel statement.
     *
     * @param anEsterelModule
     *                the esterel module which becomes a KIEL Statechart
     * @param isFinalState
     *                is true if the new state has to be a final state
     * @param theLocalEvents
     *                Stack with the local events
     * @param theLocalVariables
     *                Stack with all local variables
     * @param theTreesTrapSignals
     *Stack with all tree trap signals.
     * @throws Esterel2EstudioException
     *                if the convertion to Kiel is not working. <br>
     *                Sideeffects: none
     * @return a State
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     * @see kiel.dataStructure.State
     */
    public abstract kiel.dataStructure.State convertToKiel(
           final EsterelModule anEsterelModule,
           final boolean isFinalState,
           final ArrayList theLocalEvents,
           final ArrayList theLocalVariables,
           final ArrayList theTreesTrapSignals)
           throws Esterel2EstudioException;
     /**
     * Returns the name of an esterel statement. <br>
     * Sideeffects: none
     *
     * @return String name of the esterel statement
     */
    public final String getEsterelStatementName() {
        return this.getName()
                + this.getXMLID();
    }
    /**
     * Returns the XPath string which leads in a
     * <code>org.jdom.Document</code> representation of an .exp file to
     * the esterel statement. <br>
     * Sideeffects: none
     *
     * @return String name of the esterel statement
     * @see org.jdom.Document
     */
    public final String getXPathSearchString() {
        return this.myXPathSearchPath;
    }

    /**
     * Must be implemented in the subclasses. Parses an esterel statement
     * and set values. <br>
     * Sideeffects: see at subclasses
     *
     * @param theparser
     *                an intance of a EsterelParser
     * @throws EsterelParserException
     *                is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     * @see kiel.fileInterface.esterel.EsterelParser
     */
    public abstract void parseEsterelStatement(
           final EsterelParser theparser)
           throws EsterelParserException;

    /**
     * Parses an esterel statment in an <code>org.jdom.Document</code>
     * representation of an .exp file. Use this methode in subclasses
     * before <code>parseEsterelStatement</code> if you have not called
     * the constructor
     * <code> EsterelStatement(EsterelParser theparser)</code>.<br>
     * Sideeffects: the class variables are set
     *
     * @param theparser
     *                an instance of <code>EsterelParser</code>
     * @see #EsterelStatement(EsterelParser)
     * @see #parseEsterelStatement(EsterelParser)
     * @see kiel.fileInterface.esterel.EsterelParser
     * @see org.jdom.Document
     */
    public final void preParseEsterelStatement(
           final EsterelParser theparser) {
        this.setName(((theparser.getElement()).getName()));
         this.setXMLID((theparser.getElement()).getAttributeValue("id"));
        if (theparser.getAddAt() > 0) {
            this.myXPathSearchPath = "//*"
                    + "[local-name(.)='" + this.getName() + "'][@id='"
                    + this.getXMLID() + "']/*";
        } else {
            this.myXPathSearchPath = theparser.gettheElementXPathSearchString()
                    + "[local-name(.)='"
                    + this.getName()
                    + "'][@id='"
                    + this.getXMLID() + "']/*";
        }
    }
    /**
     * Fills the substatements in the esterel module. <br>
     * Sideeffects: none
     *
     * @param theparser
     *                the actual parser
     * @throws EsterelParserException
     *                is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     * @see kiel.fileInterface.esterel.EsterelParser
     */
    public abstract void setSubStatments(
           final EsterelParser theparser)
           throws EsterelParserException;

    /**
```

```
     * Returns a string representation of an esterel statement.
     *
     * @param anEsterelModule      an esterel modul representation
     *
     * @return a String representation of an esterel statment
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    public abstract String toString(
            final EsterelModule anEsterelModule);
} // end of class EsterelStatement
```

C.2.23. EsterelTaskDeclaration

Der Klassenaufbau ist in der Abbildung C.2.23 dargestellt.

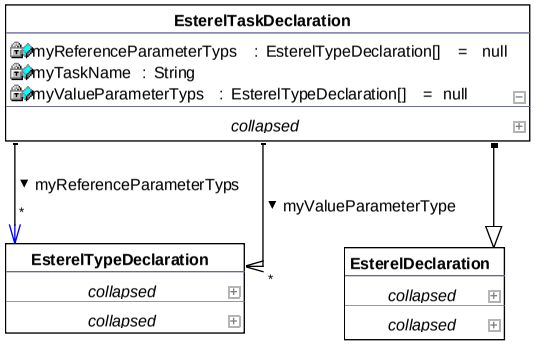


Abbildung C.18.: Klassediagramm `EsterelTaskDeclaration`

Auflistung C.28: Die Klasse `EsterelTaskDeclaration`

```
package kiel.fileInterface.esterel.esterel2estudio;

import java.util.ArrayList;
import java.util.List;

import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;

import org.jdom.Element;

/**
 * <p>
 *  It represents esterel Tasks .
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.19 $ last modified $Date: 2006/02/06 18:35:29 $
 *               <br>
 */
public class EsterelTaskDeclaration extends EsterelDeclaration {
    /**
     * the reference parameters.
     */
    private EsterelTypeDeclaration[] myReferenceParameterTyps = null;
    /**
     * the Task name.
     */
    private String myTaskName;

    /**
     * the value parameters.
     */
    private EsterelTypeDeclaration[] myValueParameterTyps = null;

    /**
     * the simple constructor.
     */
    public EsterelTaskDeclaration() {
```

```
        super();
        this.myReferenceParameterTyps = null;
        this.myValueParameterTyps = null;
        this.myTaskName = "";
    }

    /**
     * Creates an EsterelTask and fills the class variables. <br>
     * Sideeffects: class variables are changed.
     *
     * @param theparser
     *                the esterel parser
     * @param theXPath
     *                the XPath to the EsterelVariable
     * @throws EsterelParserException
     *                if something goes wrong at the parsing.
     */
    public EsterelTaskDeclaration(final EsterelParser theparser,
            final String theXPath)
            throws EsterelParserException {
        super();

        this.myReferenceParameterTyps = null;
        this.myValueParameterTyps = null;
        this.myTaskName = "";
        if (theXPath != null
                && theparser != null) {
            this.parseEsterelStatement(theparser,
                theXPath);
        } else {
            throw new EsterelParserException(
                    "TaskDelclaration : no parser or path");
        }
    }

    /**
     * @return Returns the myTaskName.
     */
    public final String getTaskName() {
        return this.myTaskName;
    }

    /**
     * Parses a Task. <br>
     * Sideeffects: sets the class variables.
     *
     * @param theXPath
     *                the XPath which leads to the Expression
     * @param theparser
     *                the esterel parser
     * @throws EsterelParserException
     *                if there is a parsing problem
     * @see kiel.fileInterface.esterel.EsterelParserException
     */
    public final void parseEsterelStatement(
            final EsterelParser theparser,
            final String theXPath)
            throws EsterelParserException {
        List anElementList = DOMHelpers.getElements(theXPath,
            theparser.getXMLDocument());
        int theEOVCounter = 0;
        int i = 0;
        if (anElementList.size() > 0) {
            this.setName(((Element) anElementList.get(0))
                .getParentElement().getName());
            this.setXMLID(((Element) anElementList.get(0))
                .getParentElement().getAttributeValue("id"));
            }
        ArrayList theReferenceParameterTyps = new ArrayList();
        ArrayList theValueParameterTyps = new ArrayList();
        while (theEOVCounter < 2
                && i < anElementList.size()) {
            Element anElement = (Element) anElementList.get(i);
            if (anElement.getName().compareTo("EOV") == 0) {
                theEOVCounter++;
            } else if (i == 0) { // parse name}
                this.myTaskName = anElement.getText();
            } else if (theEOVCounter == 0) {
                theReferenceParameterTyps.add(theparser
                    .getEsterelModule(theparser.getactualEsterModule())
                    .getEsterelTypeByID(((Element) anElementList.get(i))
                        .getAttributeValue("id")));
            } else if (theEOVCounter == 1) {
               theValueParameterTyps.add(
                   theparser
                   .getEsterelModule(theparser.getactualEsterModule())
                   .getEsterelTypeByID(((Element) anElementList.get(i))
                       .getAttributeValue("id")));
            }
            i++;
        } //while
        this.myValueParameterTyps =
            new EsterelTypeDeclaration[theValueParameterTyps.size()];
        for (int j = 0; i < theValueParameterTyps.size(); j++) {
            this.myValueParameterTyps[j] =
                (EsterelTypeDeclaration) theValueParameterTyps.get(j);
        }
        this.myValueParameterTyps =
            new EsterelTypeDeclaration[theReferenceParameterTyps.size()];
        for (int j = 0; i < theReferenceParameterTyps.size(); j++) {
            this.myValueParameterTyps[j] =
                (EsterelTypeDeclaration) theReferenceParameterTyps.get(j);
        }
    }

    /**
     * Returns the String representation of the variable.
     *
     * @param anEsterelModule
     *                  an <code>EsterelModule</code>
     * @return the String representation of an EsterelVariable
     */
```

```
public final String toString(
    final EsterelModule anEsterelModule) {
  String result = this.myTaskName
      + " ( ";
  if (this.myReferenceParameterTyps != null) {
    for (int i = 0; i < this.myReferenceParameterTyps.length; i++) {
      result += this.myReferenceParameterTyps[i].getTypeName()
          + " ,";
    }
  }
  result += " ) ( ";
  if (this.myValueParameterTyps != null) {
    for (int i = 0; i < this.myValueParameterTyps.length; i++) {
      result += this.myValueParameterTyps[i].getTypeName()
          + " ,";
    }
  }
  result += " ) ";
  return result;
}

}
```

C.2.24. EsterelTypeDeclaration

Der Klassenaufbau ist in der Abbildung C.2.24 dargestellt.

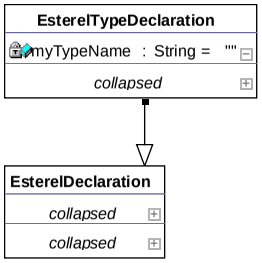


Abbildung C.19.: Klassediagramm `EsterelTypeDeclaration`

Auflistung C.29: Die Klasse `EsterelTypeDeclaration`

```
package kiel.fileInterface.esterel.esterel2estudio;


/**
 * <p>
 *  Implements a type declaration.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.3 $ last modified $Date: 2006/02/06 18:35:29 $
 */
public final class EsterelTypeDeclaration
       extends EsterelDeclaration {
    /**
     * name of the type.
     */
    private String myTypeName = "";

    /**
     * the standart constructor.
     */
    public EsterelTypeDeclaration() {
        super();
    }

    /**
     * @param aName
     *                the XML element name.
     * @param anID
     *                the XML element id
     */
    public EsterelTypeDeclaration(
           final String aName,
           final String anID) {
        super(
                aName,
                anID);
    }
    /**
     * @return Returns the myTypeName.
     */
    public String getTypeName() {
        return this.myTypeName;
    }

    /**
     * @param aTypeName
     *                The myTypeName to set.
```

```
     * @return the string representation.
     */
    public String toString() {
        return this.myTypeName;
    }
}

    */
    public void setTypeName(
            final String aTypeName) {
        this.myTypeName = aTypeName;
    }

60  /**
```

C.2.25. EsterelVariable

Der Klassenaufbau ist in der Abbildung C.2.25 dargestellt.

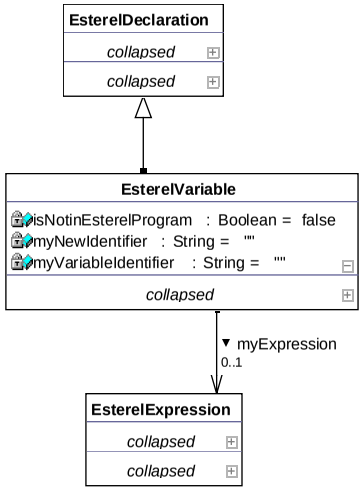


Abbildung C.20.: Klassediagramm `EsterelVariable`

Auflistung C.30: Die Klasse `EsterelVariable`

```
package kiel.fileInterface.esterel.esterel2estudio;

//
import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import kiel.dataStructure.StringVariable;
import kiel.dataStructure.Variable;
import kiel.dataStructure.boolexp.BooleanVariable;
import kiel.dataStructure.doubleexp.DoubleVariable;
import kiel.dataStructure.floatexp.FloatVariable;
import kiel.dataStructure.intexp.IntegerVariable;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import org.jdom.Element;

/**
 * <p>
 *  It represents local esterel variables .
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.38 $ last modified $Date: 2006/02/07 15:39:35 $
 *              <br>
```

```
 */
public class EsterelVariable
       extends EsterelDeclaration {
   /**
    * If it is a new variable.
    */
   private boolean isNotinEsterelProgram = false;
   /**
    * The type of the signal.(Optional).
    */
   private EsterelTypeDeclaration myChannelType; // child1

   /**
    * The expression of the signal.(Optional).
    */
   private EsterelExpression myExpression; //child
                                                                  // 2
/**
 * The new identifier name.
 */
   private String myNewIdentifier = ""; // child 0
/**
 * Me as Variable.
 */
   private Variable mySelfasVariable = null;

   /**
    * The name of the signal.
    */
   private String myVariableIdentifier = ""; // child 0

   /**
    * the standart constructor.
    */
   public EsterelVariable() {
       super();
   }
/**
 *
 * @param theparser a<code> EsterelParser</code>
 * @param anElement an element
 * @throws EsterelParserException the Exception
 */
   public EsterelVariable(final EsterelParser theparser,
          final Element anElement)
          throws EsterelParserException {
       this(
              theparser, "//*[@id='"
                     + anElement.getAttributeValue("id") + "']/*");
       this.setName(anElement.getName());
       this.setXMLID(anElement.getAttributeValue("id"));
   }

   /**
    * Creates an EsterelVariable and fills the class variables. <br>
    * Sideeffects: class variables are changed.
    *
    * @param theparser
    *                the esterel parser
    * @param theXPath
    *                the XPath to the EsterelVariable
    * @throws EsterelParserException
    *                if something goes wrong at the parsing.
    */
   public EsterelVariable(final EsterelParser theparser,
          final String theXPath)
          throws EsterelParserException {
       super();
       this.myVariableIdentifier = null;
       this.myExpression = null;
       this.myChannelType = null;
       if (theparser != null) {
           parseEsterelStatement(theparser,
               theXPath);
       }
   }

   /**
    * Returns a variable representation of the esterel variable.
    * @param anEsterelModule the esterel module which
    * becomes a KIEL Statechart
    * @param theLocalEvents Stack with the local events
    * @param theLocalVariables Stack with all local variables
    * @throws Esterel2EstudioException
    *                if the convertion to Kiel is not working. <br>
    *                Sideeffects: none
    * @return a State
    * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
    * @see kiel.dataStructure.State
    */
   public final Variable convertToKiel(
          final EsterelModule anEsterelModule,
          final ArrayList theLocalEvents,
          final ArrayList theLocalVariables)
          throws Esterel2EstudioException {
       this.createNewIdentifierName(anEsterelModule);
       Variable result = this.mySelfasVariable;
       if (result == null) {
           if (this.myChannelType != null) {
            if (this.myChannelType
                   .getTypeName()
                   .compareTo("string") == 0) {
               if (this.myExpression != null) {
                   result = new StringVariable(
                          this.getVariableIdentifier(),
                          this.myExpression
                          .convertToKielString(anEsterelModule));
               } else {
                   result = new StringVariable(
                          this.getVariableIdentifier());
               }
           } else
```

```
if (this.myChannelType
        .getTypeName()
        .compareTo("integer") == 0) {
    if (this.myExpression != null) {
        result = new IntegerVariable(
                this.myVariableIdentifier,
      this.myExpression.convertToKiel(
          anEsterelModule,
          theLocalEvents,
          theLocalVariables));

    } else {
        result = new IntegerVariable(
                this.myVariableIdentifier);
    }
} else
    if (this.myChannelType
            .getTypeName()
            .compareTo("boolean") == 0) {
        if (this.myExpression != null) {
            result = new BooleanVariable(
                    this.myVariableIdentifier,
                    this.myExpression.convertToKielBoolExp(
                        anEsterelModule,
                        theLocalEvents,
                        theLocalVariables)
              );
            } else {
                result = new BooleanVariable(
                        this.myVariableIdentifier);
            }
        } else
        if (this.myChannelType
                .getTypeName()
                .compareTo("float") == 0) {
            if (this.myExpression != null) {
                result = new FloatVariable(
                        this.myVariableIdentifier);
              /*After full suport for Float
               * this.myExpression
              .convertToKiel(anEsterelModule));
              */
            } else {
                result = new FloatVariable(
                        this.myVariableIdentifier);
            }
        } else
            if (this.myChannelType
                    .getTypeName()
                    .compareTo("double") == 0) {
                if (this.myExpression != null) {
                    result = new DoubleVariable(
                            this.myVariableIdentifier);
                            /*After full suport for double:
                  /*this.myExpression
                  .convertToKiel(anEsterelModule));*/
                } else {
                    result = new DoubleVariable(
                            this.myVariableIdentifier);
                }
            } else {
                if (this.myExpression != null) {
                     result = new StringVariable(
                            this.getVariableIdentifier(),
                            this.myExpression
                            .convertToKielString(anEsterelModule),
                            this.myChannelType.getTypeName());
                } else {
                    result = new StringVariable(
                        this.getVariableIdentifier(),
                        null,
                        this.myChannelType.getTypeName());
                }

                /*            throw new Esterel2EstudioException(
                    this.getName()
                            + "Unknown type : " + this.myChannelType);*/
                }
        } else {
            throw new Esterel2EstudioException(
                    this.getName()
                            + ": no Type");
        }
        this.mySelfasVariable = result;
    }
    return result;
}

/**
 * Creates a new name for a trap signal, the new name is the preName
 * form the preferences the name the postName of the preferences and
 * eventually a random number.
 *
 * @param anEsterelModule
 *                the esterel Module
 */
public final void createNewIdentifierName(
        final EsterelModule anEsterelModule) {
    if (this.isNotinEsterelProgram) {
        String preName = Esterel2EstudioProperties.getPreSting();
        String postName = Esterel2EstudioProperties.getPostString();
        if (preName == null) {
            preName = "";
        }
        if (postName == null) {
            postName = "";
        }
        Random rand = new Random();
        this.myNewIdentifier = preName + this.myVariableIdentifier
                + postName;
        while (anEsterelModule
                .getEsterelVariableByOrginalName(this.myNewIdentifier)
```

```
                != null) {
            this.myNewIdentifier = preName + this.myVariableIdentifier
                    + postName
                    + rand.nextInt();
        }
    }
}

/**
 * @return Returns the myChannelType.
 */
public final EsterelTypeDeclaration getChannelType() {
    return this.myChannelType;
}

/**
 * @return Returns the myExpression.
 */
public final EsterelExpression getExpression() {
    return this.myExpression;
}

/**
 * USE CAREFULLY.
 *
 * @return the orginalVariableIdentifier.
 */
public final String getOrginalVariableIdentifier() {
    return this.myVariableIdentifier;
}
/**
 *
 * @return the variable identifier.
 */
public final String getVariableIdentifier() {
    if (this.myNewIdentifier == "") {
        return this.myVariableIdentifier;
    }
    return this.myNewIdentifier;
}

/**
 * Parses a variable. <br>
 * Sideeffects: sets the class variables.
 *
 * @param theXPath
 *                the XPath which leads to the Expression
 * @param theparser
 *                the esterel parser
 * @throws EsterelParserException
 *                if there is a parsing problem
 * @see kiel.fileInterface.esterel.EsterelParserException
 */
public final void parseEsterelStatement(
        final EsterelParser theparser,
        final String theXPath)
    throws EsterelParserException {
List anElementList = DOMHelpers.getElements(theXPath,
    theparser.getXMLDocument());
if (anElementList.size() > 0
        && ((Element) anElementList.get(0)).getParentElement()
                .getName()
                .compareTo("VariableSymbol") != 0) {
    throw new EsterelParserException(
            "EsterelVariable: no Variable found.");
}
for (int i = 0; i < anElementList.size(); i++) {
    if (i == 0
            && ((Element) anElementList.get(i)).getName() != "NULL") {
        this.myVariableIdentifier = new String(
                ((Element) anElementList.get(i)).getText());
    } else if (i == 1
            && ((Element) anElementList.get(i)).getName() != "NULL") {
        this.myChannelType =
            theparser
            .getEsterelModule(theparser.getactualEsterModule())
            .getEsterelTypeByID(
                ((Element) anElementList.get(i))
                .getAttributeValue("id"));
    } else if (i == 2
            && ((Element) anElementList.get(i)).getName() != "NULL") {
        this.myExpression = new EsterelExpression(
            theparser,
                theXPath
                        + "[local-name(.)='"
                        + ((Element) anElementList.get(i)).getName()
                        + "'][@id='"
                        + ((Element) anElementList.get(i))
                        .getAttributeValue("id")
                        + "']/*", theparser.getXMLDocument());
    } else if (i == 0) {
        throw new EsterelParserException(
                "EsterelVariable:"
                        + " No Variablename");
    }
}
}

/**
 * @param aChannelType
 *                The ChannelType to set.
 */
public final void setChannelType(
        final EsterelTypeDeclaration aChannelType) {
    this.myChannelType = aChannelType;
}

/**
 * @param aExpression
 *                The Expression to set.
 */
public final void setExpression(
```

```
            final EsterelExpression aExpression) {
        this.myExpression = aExpression;
    }

    /**
     * @param aVariableIdentifier
     *                The myVariableIdentifier to set.
     */
    public final void setVariableIdentifier(
            final String aVariableIdentifier) {
        this.myVariableIdentifier = aVariableIdentifier;
    }

    /**
     * Creats a string representation of the <code>EsterelVariable</code>.
     *
     * @param anEsterelModule
     *                a <code>EsterelModule</code>
     * @return <code>this.toString()</code>
     * @see EsterelModule
     */
    public final String toString(
            final EsterelModule anEsterelModule) {
        String result = this.getVariableIdentifier();
        if (this.myExpression != null) {
            result += " := "
                    + this.myExpression.toString(anEsterelModule);
        }
        if (this.myChannelType != null) {
            result += " : "
                    + this.myChannelType;
        }
        return result
                + "\n";
    }
    /**
     * @return Returns the isNotinEsterelProgram.
     */
    public final boolean isNotinEsterelProgram() {
        return this.isNotinEsterelProgram;
    }
    /**
     * @param notinEsterelProgram The isNotinEsterelProgram to set.
     */
    public final void setNotinEsterelProgram(
            final boolean notinEsterelProgram) {
        this.isNotinEsterelProgram = notinEsterelProgram;
    }

}
```

C.2.26. EveryEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.26 dargestellt.

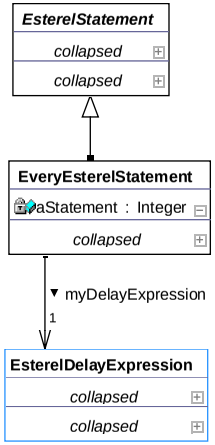


Abbildung C.21.: Klassediagramm `EveryEsterelStatement`

Auflistung C.31: Die Klasse `EveryEsterelStatement`

```
package kiel.fileInterface.esterel.esterel2estudio;
import java.util.ArrayList;
import java.util.List;
//
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.Node;
import kiel.dataStructure.SimpleState;
import kiel.dataStructure.State;
import kiel.dataStructure.Transition;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StateChartHelpers;
//
import org.jdom.Element;
/**
 * <p> This is a subclass of <code>EsterelStatement</code>
 * classes implements the every statement.</p>
 * <p>Copyright: Copyright (c) 2005</p>
 * <p>Company: Uni Kiel</p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
 * @version $Revision: 1.32 $ last modified $Date: 2006/02/17 20:00:22 $
 */
public class EveryEsterelStatement extends EsterelStatement {
    /** The index number of the substatement in an
     * <code>EsterelModule</code>.
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    private int aStatement;
    /**
     * The delay expression as <code>EsterelDelayExpression</code>.
     *  @see EsterelDelayExpression
     */
```

```
private EsterelDelayExpression myDelayExpression;
/**
* The standart constructor.
 */
 public EveryEsterelStatement() {
     super();
     this.aStatement = -1;
     this.myDelayExpression = null;
 }
 /**
 * Parses a Document and set the class varaibles.
 * <br>It calls also the super constuctor<code>EsterelStatement
 * (EsterelParser theparser)</code> .
 * <br>Calls the <code>parseEsterelStatement</code> methode
 * <br>Sideeffects:
 * Changes <code>EsterelParser</code>class variables
 *@param theparser an intace of a EsterelParser
 * @throws EsterelParserException is only delivered
 * @see kiel.fileInterface.esterel.EsterelParserException
 * @see #parseEsterelStatement(EsterelParser)
 * @see EsterelStatement#EsterelStatement(EsterelParser)
 * @see kiel.fileInterface.esterel.EsterelParser
 */
public EveryEsterelStatement(final EsterelParser theparser)
    throws EsterelParserException {
    super(theparser);
    this.aStatement = -1;
    this.myDelayExpression = null;
    if (theparser != null) {
        this.parseEsterelStatement(theparser);
    } else {
        throw new EsterelParserException("no parser");
    }
}
/**
 * Returns a state representation of the esterel statement.
 * @param anEsterelModule the esterel module which
 * becomes a KIEL Statechart
 *@param isFinalState is true if the new state has to be
 * a final state
 * @param theLocalEvents Stack with the local events
 * @param theLocalVariables Stack with all local variables
 * @param theTreesTrapSignals
 *                Stack with all tree trap signals.
 * @throws Esterel2EstudioException
 *                if the convertion to Kiel is not working. <br>
 *                Sideeffects: none
 * @return a State
 * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
 * @see kiel.dataStructure.State
 */
public final kiel.dataStructure.State convertToKiel(
       final EsterelModule anEsterelModule,
       final boolean isFinalState,
       final ArrayList theLocalEvents,
       final ArrayList theLocalVariables,
       final ArrayList theTreesTrapSignals)
       throws Esterel2EstudioException {
    InitialArc aInitialArc = new InitialArc();
    InitialState anInitialState = new InitialState();
    SimpleState theStartState = new SimpleState();
    State theStatementState =
       (
           (EsterelStatement)
              (
                  anEsterelModule.getEsterelProgram().get(
                     this.aStatement))).convertToKiel(
              anEsterelModule,
              false,
              theLocalEvents,
              theLocalVariables,
              theTreesTrapSignals);
    // false means, substate is not a finalstate beacause its aloop
    int priority = 1;
    Transition theLoopTrans =
       StateChartHelpers.createSA(
           theStatementState,
           priority,
           theStatementState);
    Transition theStartTrans =
       StateChartHelpers.createSA(
           theStartState,
           priority,
           theStatementState);
      theStartTrans.setLabel(
       this.myDelayExpression.convertToKiel(
           anEsterelModule,
           theLocalEvents,
           theLocalVariables,
           null));
      boolean aDummy = this.myDelayExpression.isImmediate();
      this.myDelayExpression.setImmediate(false);
theLoopTrans.setLabel(
       this.myDelayExpression.convertToKiel(
           anEsterelModule,
           theLocalEvents,
           theLocalVariables,
           null));
   this.myDelayExpression.setImmediate(aDummy);
       aInitialArc.setSource(anInitialState);
       aInitialArc.setTarget(theStartState);

           return StateChartHelpers.createOR(
               this.getEsterelStatementName() + "state",
               new Node[] {
                   anInitialState,
                   theStatementState,
                   theStartState });

   }
   /**
    *Implemtents the abstrac methode from <code>EsterelStatement</code>
```

```
     * Parses an esterel statment in an <code>org.jdom.Document</code>
     * representation of an .exp file
     * if you have not called the constructor
     * <code> EsterelStatement(EsterelParser theparser)</code>
     * use the methode <code>preParseEsterelStatement</code>
     * before <code>parseEsterelStatement</code>.
     * <br>Sideeffects:
     * the class variables are set
     * Changes <code>EsterelParser</code>class variables
     * @param theparser an instance of <code>EsterelParser</code>
     * @throws EsterelParserException is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     * @see EsterelStatement#EsterelStatement(EsterelParser)
     * @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see kiel.fileInterface.esterel.EsterelParser
     * @see org.jdom.Document
     */
    public final void parseEsterelStatement(final EsterelParser theparser)
        throws EsterelParserException {
        List anElementList = null;
        final int index = 1;
        anElementList =
            DOMHelpers.getElements(
                theparser.getXMLDocument(),
                this.getXPathSearchString());
        if (anElementList.size() == 2) {
            theparser.incStatementCounter();
            this.myDelayExpression =
                new EsterelDelayExpression(
                    theparser,
                    "//*[local-name(.)='"
                        + ((Element) anElementList.get(index)).getName()
                        + "'][@id='"
                        + ((Element) anElementList.get(index))
                            .getAttributeValue("id")
                        + "']/*",
                    theparser.getXMLDocument());
        } else {
            throw new EsterelParserException(
                this.getEsterelStatementName() + " statement wrong ",
                theparser);
        } //else if size
    }
    /**
     * Fills the substatements in the esterel module.
     * In this class it is fills the do Statements if there are any
     *<br> Sideeffects: none
     * @param theparser the actual parser
     * @throws EsterelParserException is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     *  @see kiel.fileInterface.esterel.EsterelParser
     */
    public final void setSubStatments(final EsterelParser theparser)
        throws EsterelParserException {
        int i = 0;
        List anElementList = null;
        anElementList =
            DOMHelpers.getElements(
                theparser.getXMLDocument(),
                this.getXPathSearchString());
        if (anElementList.size() == 2) {
            this.aStatement = theparser.getAddAt();
            theparser.getEsterelModule(
                theparser.getactualEsterModule())
                .addStatementToModule(
                    theparser.getAddAt(),
                    theparser.createStatement(((Element) (anElementList.get(i)))));
        }
    }
    /**
     *Implementents the abstrac methode from <code>EsterelStatement</code>.
     * Returns a string representation of an esterel statement.
     * @param anEsterelModule an esterel modul representation
     * @return a String representation of an esterel statment
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
     *@see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    public final String toString(final EsterelModule anEsterelModule) {
        return this.getEsterelStatementName()
            + " "
            + this.myDelayExpression.toString(anEsterelModule)
            + " do \n"
            + (
                (EsterelStatement) anEsterelModule.getEsterelProgram().get(
                    this.aStatement)).toString(
                anEsterelModule)
            + "end "
            + this.getEsterelStatementName();
    }
}
```

C.2.27. ExitEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.27 dargestellt.

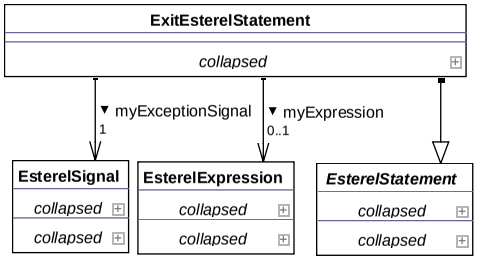


Abbildung C.22.: Klassediagramm `ExitEsterelStatement`

Auflistung C.32: Die Klasse `ExitEsterelStatement`

```
package kiel.fileInterface.esterel.esterel2estudio;

//
import java.util.ArrayList;
import java.util.List;

import kiel.dataStructure.CompoundLabel;
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.Node;
import kiel.dataStructure.SimpleState;
import kiel.dataStructure.StringLabel;
import kiel.dataStructure.TransitionLabel;
import kiel.dataStructure.eventexp.Event;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StateChartHelpers;

import org.jdom.Element;

/**
 * <p>
 *  This is a subclass of <code>EsterelStatement</code>
 * classes implements the exit statement.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2004
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.56 $ last modified $Date: 2006/02/06 18:35:29 $
 */
public class ExitEsterelStatement
      extends EsterelStatement {
   /** The expression of the signal to exit. */
   private EsterelExpression myExpression;

   /** The exception signal to exit. */
   private EsterelSignal myExceptionSignal = null;

   /**
    * Simple constructor.
    */
   public ExitEsterelStatement() {
      super();
   } //public ExitEsterelStatement()

   /**
    * Parses a Document and set the class varaibles. <br>
    * It calls also the super constuctor <code>EsterelStatement
```

```
 * (EsterelParser theparser)</code>.
 * <br>
 * Calls the <code>parseEsterelStatement</code> methode <br>
 * Sideeffects: Changes <code>EsterelParser</code> class variables
 *
 * @param theparser
 *                an intace of a EsterelParser
 * @throws EsterelParserException
 *                is only delivered
 * @see kiel.fileInterface.esterel.EsterelParserException
 * @see EsterelStatement#parseEsterelStatement(EsterelParser)
 * @see EsterelStatement#EsterelStatement(EsterelParser)
 * @see kiel.fileInterface.esterel.EsterelParser
 */
public ExitEsterelStatement(final EsterelParser theparser)
      throws EsterelParserException {
   super(
         theparser);
   this.parseEsterelStatement(theparser);
} //public ExitEsterelStatement(final EsterelParser theparser)

/**
 * Implements the methode from <code>EsterelStatement</code>.
 * Returns a state representation of the esterel statement
 *
 * @param anEsterelModule
 *                the esterel module which becomes a KIEL Statechart
 * @param isFinalState
 *                is true if the new state has to be a final state
 * @param theLocalEvents
 *                Stack with the local events
 * @param theLocalVariables
 *                Stack with all local variables <br>
 *                Sideeffects: none
 * @param theTreesTrapSignals
 *                Stack with all tree trap signals.
 * @throws Esterel2EstudioException
 *                if the convertion to Kiel is not working. <br>
 *                Sideeffects: none
 * @return a State
 * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
 * @see kiel.dataStructure.State
 */
public final kiel.dataStructure.State convertToKiel(
      final EsterelModule anEsterelModule,
      final boolean isFinalState,
      final ArrayList theLocalEvents,
      final ArrayList theLocalVariables,
      final ArrayList theTreesTrapSignals)
      throws Esterel2EstudioException {
   InitialArc aInitialArc = new InitialArc();
   InitialState anInitialState = new InitialState();
   SimpleState theEndState = new SimpleState();
   aInitialArc.setSource(anInitialState);
   aInitialArc.setTarget(theEndState);
   boolean found = false;
   String myName = this.myExceptionSignal.getSignalIdentifier();
   String emitthese = myName;
   if (this.myExpression != null) {
      emitthese += "(" + this.myExpression.toString(anEsterelModule)
            + ")";
   }
   for (int i = theTreesTrapSignals.size() - 1; i > 0 && !found; i--) {
      if (theTreesTrapSignals.get(i) instanceof TrapEsterelStatement) {
      TrapEsterelStatement aTrap =
          (TrapEsterelStatement) theTreesTrapSignals.get(i);
      EsterelSignal[] allExceptionDeclaration =
          aTrap.getMyExceptionDeclarationList();
      for (int j = allExceptionDeclaration.length - 1; j > -1; j--) {
          if (allExceptionDeclaration[j].getSignalIdentifier()
                  .compareTo(myName) == 0) {
              found = true;
          } else {
              emitthese += ", " + aTrap.getMyHaltSignal()
                      .getSignalIdentifier();
          }
      }
      } else {
          AbortEsterelStatement anAbort =
              (AbortEsterelStatement) theTreesTrapSignals.get(i);
          emitthese += ", " + anAbort.getMyHaltSignal()
          .getSignalIdentifier();
      }
   } //for
   TransitionLabel theLabelofaTranstion = new CompoundLabel();
   Event anEvent = this.myExceptionSignal.convertToKiel(anEsterelModule);
   if (anEvent != null) {
      ((CompoundLabel) theLabelofaTranstion)
      .setEffect(StateChartHelpers
            .getActions(anEsterelModule.getEsterelStateChart(),
          theLocalVariables,
          theLocalEvents,
          emitthese));
      if (((CompoundLabel) theLabelofaTranstion).getEffect() == null) {
          theLabelofaTranstion = new StringLabel(
                  emitthese);
      }
   }
   aInitialArc.setLabel(theLabelofaTranstion);
   if (isFinalState) {
      return StateChartHelpers
      .createFinalOR(this.getEsterelStatementName()
            + "state",
          new Node[] {
                  anInitialState, theEndState });
   }
   return StateChartHelpers
   .createOR(this.getEsterelStatementName()
          + "state",
      new Node[] {
              anInitialState, theEndState });
} //public final kiel.dataStructure.State convertToKiel(
```

```
/**
 * Implemtents the abstrac methode from <code>EsterelStatement</code>.
 * Parses an esterel statment in an <code>org.jdom.Document</code>
 * representation of an .exp file. if you have not called the
 * constructor <code> EsterelStatement(EsterelParser theparser)</code>
 * use the methode <code>preParseEsterelStatement</code> before
 * <code>parseEsterelStatement</code><br>
 * Sideeffects: the class variables are set Changes
 * <code>EsterelParser</code> class variables
 *
 * @param theparser
 *                an instance of <code>EsterelParser</code>
 * @throws EsterelParserException
 *                is only delivered
 * @see kiel.fileInterface.esterel.EsterelParserException
 * @see EsterelStatement#EsterelStatement(EsterelParser)
 * @see EsterelStatement#parseEsterelStatement(EsterelParser)
 * @see EsterelStatement#preParseEsterelStatement(EsterelParser)
 * @see kiel.fileInterface.esterel.EsterelParser
 * @see org.jdom.Document
 */
public final void parseEsterelStatement(
       final EsterelParser theparser)
       throws EsterelParserException {
   final int toomuch = 3;
   List anElementList = null;
   anElementList =
       DOMHelpers.getElements(theparser.getXMLDocument(),
       this.getXPathSearchString());
   if (anElementList.size() > 0 && anElementList.size() < toomuch) {
       this.myExceptionSignal =
           theparser
               .getEsterelModule(theparser.getactualEsterModule())
               .getEsterelSignalByID(
                   this.getSignal(anElementList));
       if (anElementList.size() > 1 && anElementList.size() < toomuch) {
           final int theIndex = 1;
           this.myExpression = new EsterelExpression(
               theparser,
                   this.getXPathSearchString() + "[local-name(.)='"
                           + ((Element) anElementList.get(theIndex))
                           .getName()
                           + "'][@id='"
                           + ((Element) anElementList.get(theIndex))
                           .getAttributeValue("id")
                           + "']/*", theparser.getXMLDocument());
       } //if >1
   } else {
       throw new EsterelParserException(
               this.getEsterelStatementName() + " no signal expression",
               theparser);
   } //if >0
}

/**
 * Fills the substatements in the esterel module in this class it is
 * empty. <br>
 * Sideeffects: none
 *
 * @param theparser
 *                the actual parser
 * @throws EsterelParserException
 *                is only delivered
 * @see kiel.fileInterface.esterel.EsterelParserException
 * @see kiel.fileInterface.esterel.EsterelParser
 */
public final void setSubStatments(
       final EsterelParser theparser)
       throws EsterelParserException {
}

/**
 * Implemtents the abstrac methode from <code>EsterelStatement</code>.
 * Returns a string representation of an esterel statement.
 *
 * @param anEsterelModule
 *                an esterel modul representation
 * @return a String representation of an esterel statment
 * @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
 * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
 */
public final String toString(
       final EsterelModule anEsterelModule) {
   return this.getEsterelStatementName() + " "
           + this.myExceptionSignal.getSignalIdentifier()
           + " "
           + this.myExpression
           + "";
}
/**
 * Retuns the signal of the Signalexpession. <br>
 * Sideeffects: none
 *
 * @param anElementList
 *                the cildren elements of the SignalExpression
 * @return the signals id
 * @throws EsterelParserException
 *                if no signal is found
 */
private String getSignal(
       final List anElementList)
       throws EsterelParserException {
   final int theSignalIndex = 0;
   if (anElementList.size() > theSignalIndex) {
       return ((Element) anElementList.get(theSignalIndex))
       .getAttributeValue("id");
   }
   throw new EsterelParserException(
         this.getEsterelStatementName() + ": found no signal ");
}
} // end of class
```

C.2.28. HaltEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.28 dargestellt.

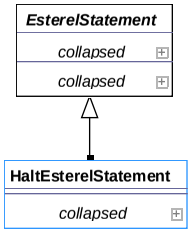


Abbildung C.23.: Klassediagramm `HaltEsterelStatement`

Auflistung C.33: Die Klasse `HaltEsterelStatement`

```
package kiel.fileInterface.esterel.esterel2estudio;
import java.util.ArrayList;
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.Node;
import kiel.dataStructure.SimpleState;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.StateChartHelpers;
//
/**
  * <p> This is a subclass of <code>EsterelStatement</code>
 * classes implements the nothing statement.</p>
 * <p>Copyright: Copyright (c) 2004</p>
 * <p>Company: Uni Kiel</p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
 * @version $Revision: 1.37 $ last modified $Date: 2006/02/06 18:35:29 $
 * <br>
 * $Log: HaltEsterelStatement.java,v $
 * Revision 1.37 2006/02/06 18:35:29 lku
 * *** empty log message ***
 *
 * Revision 1.35 2005/11/10 12:26:26 lku
 * *** empty log message ***
 *
 * Revision 1.27 2005/10/04 11:33:29 lku
 * *** empty log message ***
 *
 * Revision 1.7 2005/05/12 01:37:57 lku
 * *** empty log message ***
 *
 * Revision 1.8 2005/01/14 16:42:26 lku
 * *** empty log message ***
 *
 * <br>
 * Revision 1.7 2005/01/12 15:00:00 lku
 * <br>
 * add javadoc + check for project manual code conventions
 * <br>
 * Revision 1.8 2005/01/14 15:00:00 lku
 * <br>
 * fix javadoc
*/
public class HaltEsterelStatement extends EsterelStatement {
    /**
     * Simple constructor.
     */
    public HaltEsterelStatement() {
        super();
    }
    /**
    * Parses a Document and set the class varaibles.
    * <br>It calls also the super constuctor<code>EsterelStatement
    * (EsterelParser theparser)</code> .
    * <br>Calls the <code>parseEsterelStatement</code> methode
    * <br>Sideeffects:
```

```
     * Changes <code>EsterelParser</code>class variables
     * @param theparser an intace of a EsterelParser
     * @see #parseEsterelStatement(EsterelParser)
     * @see EsterelStatement#EsterelStatement(EsterelParser)
     * @see kiel.fileInterface.esterel.EsterelParser
     */
    public HaltEsterelStatement(final EsterelParser theparser) {
        super(theparser);
        this.parseEsterelStatement(theparser);
    } //public NothingEsterelStatement(final EsterelParser theparser) {
    /**
    *Implements the methode from <code>EsterelStatement</code>.
    *  Returns a state representation of the esterel statement
    * @param anEsterelModule the esterel module which
    * becomes a KIEL Statechart
    *@param isFinalState is true if the new state has to be
     * a final state
     * @param theLocalEvents Stack with the local events
     * @param theLocalVariables Stack with all local variables
    * <br>Sideeffects:
    * none
    ** @param theTreesTrapSignals
     *                Stack with all tree trap signals.
     * @throws Esterel2EstudioException
     *                if the convertion to Kiel is not working. <br>
     *                Sideeffects: none
     * @return a State
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     * @see kiel.dataStructure.State
     */
    public final kiel.dataStructure.State convertToKiel(
           final EsterelModule anEsterelModule,
           final boolean isFinalState,
           final ArrayList theLocalEvents,
           final ArrayList theLocalVariables,
           final ArrayList theTreesTrapSignals)
           throws Esterel2EstudioException {
        InitialState anInitialState = new InitialState();
        InitialArc anArc = new InitialArc();
        SimpleState theHaltState = new SimpleState();
        anArc.setSource(anInitialState);
        anArc.setTarget(theHaltState);
        return StateChartHelpers.createOR(this.getEsterelStatementName()
               + "_state", new Node[] {anInitialState, theHaltState });
    } //public final kiel.dataStructure.State convertToKiel(
    /**
    *Implemtents the abstrac methode from <code>EsterelStatement</code>
    * Parses an esterel statment in an <code>org.jdom.Document</code>
    * representation of an .exp file
    *  if you have not called the constructor
    * <code> EsterelStatement(EsterelParser theparser)</code>
    * use the methode <code>preParseEsterelStatement</code>
    * before <code>parseEsterelStatement</code>
    * In this special case nothing has to be done.
     *  <br>Sideeffects:
    * none
    * Changes <code>EsterelParser</code>class variables
    * @param theparser an instance of <code>EsterelParser</code>
    * @see EsterelStatement#EsterelStatement(EsterelParser theparser)
    *  @see EsterelStatement#parseEsterelStatement(EsterelParser)
    * @see kiel.fileInterface.esterel.EsterelParser
    * @see org.jdom.Document
    */
    public final void parseEsterelStatement(final EsterelParser theparser) { }
    /**
     * Fills the substatements in the esterel module
     * in this class it is empty.
     *<br> Sideeffects: none
     * @param theparser the actual parser
    * @throws EsterelParserException is only delivered
    * @see kiel.fileInterface.esterel.EsterelParserException
     *  @see kiel.fileInterface.esterel.EsterelParser
     */
    public final void setSubStatments(final EsterelParser theparser)
        throws EsterelParserException { }
    /**
     *Implemtents the abstrac methode from <code>EsterelStatement</code>.
    * Returns a string representation of an esterel statement
    * @param anEsterelModule an esterel modul representation
    * @return a String representation of an esterel statment
    * @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
    *@see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
    */
    public final String toString(final EsterelModule anEsterelModule) {
        return this.getEsterelStatementName() + "\n";
    }
}
```

C.2.29. IfEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.29 dargestellt.

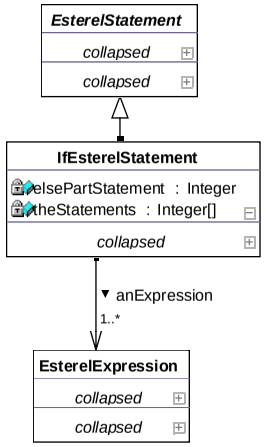


Abbildung C.24.: Klassediagramm `IfEsterelStatement`

Auflistung C.34: Die Klasse `IfEsterelStatement`

```
package kiel.fileInterface.esterel.esterel2estudio;
//
import java.util.ArrayList;
import java.util.List;
import kiel.dataStructure.ConditionalTransition;
import kiel.dataStructure.DynamicChoice;
import kiel.dataStructure.FinalSimpleState;
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.Node;
import kiel.dataStructure.State;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StateChartHelpers;
import org.jdom.Element;
/**
 * <p>This is a subclass of <code>EsterelStatement</code>
 * classes implements the if statement.</p>
 * <p>Copyright: Copyright (c) 2004</p>
 * <p>Company: Uni Kiel</p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
 * @version $Revision: 1.38 $ last modified $Date: 2006/02/06 18:35:29 $
 */
public class IfEsterelStatement extends EsterelStatement {
    /** The SignalExpressions.*/
    private EsterelExpression[] anExpression;
    /** the index number of the else part substatement in an
     * <code>EsterelModule</code>.
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    private int elsePartStatement;
    /** the index numbers of the then part substatements in an
     * <code>EsterelModule</code>.
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
```

```
*/
private int[] theStatements;
/**
 * the standart constructor.
 */
public IfEsterelStatement() {
    super();
    this.theStatements = null;
    this.elsePartStatement = -1;
    this.anExpression = null;
}
/**
* Parses a Document and set the class varaibles.
* <br>It calls also the super constuctor<code>EsterelStatement
* (EsterelParser theparser)</code> .
* <br>Calls the <code>parseEsterelStatement</code> methode
* <br>Sideeffects:
* Changes <code>EsterelParser</code>class variables
*@param theparser an intace of a EsterelParser
* @throws EsterelParserException is only delivered
* @see kiel.fileInterface.esterel.EsterelParserException
* @see #parseEsterelStatement(EsterelParser)
* @see EsterelStatement#EsterelStatement(EsterelParser)
* @see kiel.fileInterface.esterel.EsterelParser
*/
public IfEsterelStatement(final EsterelParser theparser)
    throws EsterelParserException {
    super(theparser);
    this.theStatements = null;
    this.elsePartStatement = -1;
    this.anExpression = null;
    if (theparser != null) {
        this.parseEsterelStatement(theparser);
    }
}
/**
 * Returns a state representation of the esterel statement.
 * @param anEsterelModule the esterel module which
 * becomes a KIEL Statechart
 *@param isFinalState is true if the new state has to be
 * a final state
 * @param theLocalEvents Stack with the local events
 * @param theLocalVariables Stack with all local variables
 * @param theTreesTrapSignals
 *                Stack with all tree trap signals.
 * @throws Esterel2EstudioException
 *                if the convertion to Kiel is not working. <br>
 *                Sideeffects: none
 * @return a State
 * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
 * @see kiel.dataStructure.State
 */
public final kiel.dataStructure.State convertToKiel(
       final EsterelModule anEsterelModule,
       final boolean isFinalState,
       final ArrayList theLocalEvents,
       final ArrayList theLocalVariables,
       final ArrayList theTreesTrapSignals)
       throws Esterel2EstudioException {
InitialArc aInitialArc = new InitialArc();
InitialState anInitialState = new InitialState();
DynamicChoice aPresentTest = new DynamicChoice();
int nrOfEndStates = this.anExpression.length + 1;
State[] theEndStates = new State[nrOfEndStates];
// the cases min 0
for (int theCaseIndex = 0;
    theCaseIndex < this.anExpression.length;
    theCaseIndex++) {
    if (this.theStatements[theCaseIndex] == -1) {
        theEndStates[theCaseIndex] = new FinalSimpleState();
    } else {
        theEndStates[theCaseIndex] =
            (
                (EsterelStatement)
                    (
                        anEsterelModule.getEsterelProgram().get(
                            this
                                .theStatements[theCaseIndex]))
                                .convertToKiel(
                    anEsterelModule,
                    true,
                    theLocalEvents,
                    theLocalVariables,
                    theTreesTrapSignals);
    }
    ConditionalTransition aChoiceTransition =
        new ConditionalTransition();
    StateChartHelpers.set(
        aChoiceTransition,
        aPresentTest,
        theCaseIndex + 1,
        theEndStates[theCaseIndex]);
    aChoiceTransition.setLabel(
        this.anExpression[theCaseIndex].convertToKiel(
            anEsterelModule,
            theLocalEvents,
            theLocalVariables,
            null));
} //for
//the default choice
if (this.elsePartStatement != -1) {
    theEndStates[nrOfEndStates - 1] =
        (
            (EsterelStatement)
                (
                    anEsterelModule.getEsterelProgram().get(
                        this.elsePartStatement))).convertToKiel(
            anEsterelModule,
            true,
            theLocalEvents,
            theLocalVariables,
            theTreesTrapSignals);
```

```
        } else {
            theEndStates[nrOfEndStates - 1] = new FinalSimpleState();
        }
        ConditionalTransition aChoiceTransition = new ConditionalTransition();
        StateChartHelpers.set(
            aChoiceTransition,
            aPresentTest,
            nrOfEndStates,
            theEndStates[nrOfEndStates - 1]);
        aInitialArc.setSource(anInitialState);
        aInitialArc.setTarget(aPresentTest);
        Node[] theNodes = new Node[theEndStates.length + 2];
        for (int i = 0; i < theNodes.length - 2; i++) {
            theNodes[i] = theEndStates[i];
        }
        theNodes[theNodes.length - 2] = anInitialState;
        theNodes[theNodes.length - 1] = aPresentTest;
        if (isFinalState) {
            return StateChartHelpers.createFinalOR(
                this.getEsterelStatementName() + "state",
                theNodes);
        }
            return StateChartHelpers.createOR(
                this.getEsterelStatementName() + "state",
                theNodes);

    }
    /**
     *Implemtents the abstrac methode from <code>EsterelStatement</code>
     * Parses an esterel statment in an <code>org.jdom.Document</code>
     * representation of an .exp file
     *  if you have not called the constructor
     * <code> EsterelStatement(EsterelParser theparser)</code>
     * use the methode <code>preParseEsterelStatement</code>
     * before <code>parseEsterelStatement</code>.
      *  <br>Sideeffects:
     * the class variables are set
     * Changes <code>EsterelParser</code>class variables
     * @param theparser an instance of <code>EsterelParser</code>
    * @throws EsterelParserException is only delivered
    * @see kiel.fileInterface.esterel.EsterelParserException
     * @see EsterelStatement#EsterelStatement(EsterelParser)
     *  @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see kiel.fileInterface.esterel.EsterelParser
     * @see org.jdom.Document
     */
    public final void parseEsterelStatement(final EsterelParser theparser)
        throws EsterelParserException {
        List anElementList =
            DOMHelpers.getElements(
                theparser.getXMLDocument(),
                this.getXPathSearchString());
        int numberOfPredicated = 0;
        if (((Element) anElementList.get(anElementList.size() - 1)).getName()
            == "PredicatedStatement") {
            numberOfPredicated = anElementList.size();
        } else {
            numberOfPredicated = anElementList.size() - 1;
            theparser.incStatementCounter();
        }
        this.anExpression = new EsterelExpression[numberOfPredicated];
        this.theStatements = new int[numberOfPredicated];
        //set default values
        for (int i = 0; i < numberOfPredicated; i++) {
            this.anExpression[i] = null;
            this.theStatements[i] = -1;
        }
        int aCaseStatementCounter = 0;
        for (int theCaseIndex = 0;
            theCaseIndex < numberOfPredicated;
            theCaseIndex++) {
            if (((Element) anElementList.get(theCaseIndex)).getName()
                == "PredicatedStatement") {
                String aPredicatedStatementXPathSearchString =
                    this.getXPathSearchString()
                        + "[local-name(.)='"
                        + ((Element) anElementList.get(theCaseIndex)).getName()
                        + "'][@id='"
                        + (
                            (Element) anElementList.get(
                                theCaseIndex)).getAttributeValue(
                            "id")
                        + "']/*";
                List aPredicatedElementList =
                    DOMHelpers.getElements(
                        theparser.getXMLDocument(),
                        aPredicatedStatementXPathSearchString);
                if (aPredicatedElementList.size() > 1) {
                    aCaseStatementCounter++;
                }
                int index = aPredicatedElementList.size() - 1;
                // the SignalExpression is the second listelement
                String aSubPredicatedStatementXPathSearchString =
                    "//*[local-name(.)='"
                        + ((Element) aPredicatedElementList.get(index))
                        .getName()
                        + "'][@id='"
                        + (
                            (Element) aPredicatedElementList.get(
                                index)).getAttributeValue(
                            "id")
                        + "']/*";
                this.anExpression[theCaseIndex] =
                    new EsterelExpression(
                        theparser,
                        aSubPredicatedStatementXPathSearchString,
                        theparser.getXMLDocument());
            } else {
                throw new EsterelParserException(
                    this.getEsterelStatementName() + " wrong statement",
                    theparser);
```

```
            } //else if size*/
        }
        theparser.incStatementCounter(aCaseStatementCounter);
    }
    /**
    * Fills the substatements in the esterel module.
    * In this class it is fills the do Statements if there are any
    *<br> Sideeffects: none
    * @param theparser the actual parser
    * @throws EsterelParserException is only delivered
    * @see kiel.fileInterface.esterel.EsterelParserException
    *  @see kiel.fileInterface.esterel.EsterelParser
    */
    public final void setSubStatments(final EsterelParser theparser)
        throws EsterelParserException {
        List anElementList =
            DOMHelpers.getElements(
                theparser.getXMLDocument(),
                this.getXPathSearchString());
        for (int theCaseIndex = 0;
            theCaseIndex < anElementList.size();
            theCaseIndex++) {
            if (((Element) anElementList.get(theCaseIndex)).getName()
                == "PredicatedStatement") {
                String aPredicatedStatementXPathSearchString =
                    this.getXPathSearchString()
                        + "[local-name(.)='"
                        + ((Element) anElementList.get(theCaseIndex)).getName()
                        + "'][@id='"
                        + (
                            (Element) anElementList.get(
                                theCaseIndex)).getAttributeValue(
                            "id")
                        + "']/*";
                List aPredicatedElementList =
                    DOMHelpers.getElements(
                        theparser.getXMLDocument(),
                        aPredicatedStatementXPathSearchString);
                if (aPredicatedElementList.size() > 1) {
                    this.theStatements[theCaseIndex] = theparser.getAddAt();
                    theparser.getEsterelModule(
                        theparser.getactualEsterModule()).addStatementToModule(
                        theparser.getAddAt(),
                        theparser.createStatement(
                            ((Element) (aPredicatedElementList.get(0)))));
                } else {
                    this.theStatements[theCaseIndex] = -1;
                }
            } else {
                this.elsePartStatement = theparser.getAddAt();
                theparser.getEsterelModule(
                    theparser.getactualEsterModule()).addStatementToModule(
                    theparser.getAddAt(),
                    theparser.createStatement(
                        ((Element) (anElementList
                            .get(anElementList.size() - 1)))));
            }
        } //for
    }
    /**
     *Implemtents the abstrac methode from <code>EsterelStatement</code>.
    * Returns a string representation of an esterel statement.
    * @param anEsterelModule an esterel modul representation
    * @return a String representation of an esterel statment
    * @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
    *@see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
    */
    public final String toString(final EsterelModule anEsterelModule) {
        String result = this.getEsterelStatementName();
        if (this.anExpression.length > 1) {
            for (int theCaseIndex = 0;
                theCaseIndex < this.anExpression.length;
                theCaseIndex++) {
                result += "\n"
                    + "     case "
                    + this.anExpression[theCaseIndex].toString(anEsterelModule);
                if (this.theStatements[theCaseIndex] != -1) {
                    result += " do "
                        + (
                            (EsterelStatement) anEsterelModule
                                .getEsterelProgram()
                                .get(
                                this.theStatements[theCaseIndex])).toString(
                            anEsterelModule);
                }
            }
        } else {
            result += " " + this.anExpression[0];
            if (this.theStatements[0] != -1) {
                result += " then "
                    + "\n"
                    + (
                        (EsterelStatement) anEsterelModule
                            .getEsterelProgram()
                            .get(
                            this.theStatements[0])).toString(
                        anEsterelModule);
            } else {
                result += "\n";
            }
        }
        if (this.elsePartStatement > -1) {
            result += "else "
                + (
                    (EsterelStatement) anEsterelModule.getEsterelProgram().get(
                        this.elsePartStatement)).toString(
                    anEsterelModule);
        } //elsePart
        result += "\n end " + this.getEsterelStatementName();
        return result;
    }
}
```

C.2.30. LocalSignalDeclarationEsterelStatement

Auflistung C.35: Die Klasse LocalSignalDeclarationEsterelStatement

```
package kiel.fileInterface.esterel.esterel2estudio;
//
import java.util.ArrayList;
import java.util.List;
//
import kiel.dataStructure.CompositeState;
import kiel.dataStructure.FinalORState;
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.ORState;
import kiel.dataStructure.State;
import kiel.dataStructure.eventexp.Event;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StateChartHelpers;
//
import org.jdom.Element;
/**
 * <p> This is a subclass of <code>EsterelStatement</code>
 * classes implements the signal in statement.</p>
 * <p>Copyright: Copyright (c) 2004</p>
 * <p>Company: Uni Kiel</p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
 * @version $Revision: 1.2 $ last modified $Date: 2006/02/07 15:39:35 $
 * <br>
 */
public class LocalSignalDeclarationEsterelStatement extends EsterelStatement {
    /** the local signals. */
    private EsterelSignal[] theLocalSignals = null;
    /** The index number of the substatement in an
     * <code>EsterelModule</code>.
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    private int aStatement;
    /**
    *Simple constructor.
    */
    public LocalSignalDeclarationEsterelStatement() {
        super();
        this.aStatement = -1;
    } //public LocalSignalDeclarationEsterelStatement() {
    /**
    * Parses a Document and set the class varaibles.
    * <br>It calls also the super constuctor<code>EsterelStatement
    * (EsterelParser theparser)</code> .
    * <br>Calls the <code>parseEsterelStatement</code> methode
    * <br>Sideeffects:
    * Changes <code>EsterelParser</code>class variables
    * @param theparser an intace of a EsterelParser
    * @throws EsterelParserException is only delivered
    * @see kiel.fileInterface.esterel.EsterelParserException
    * @see #parseEsterelStatement(EsterelParser)
    * @see EsterelStatement#EsterelStatement(EsterelParser)
    * @see kiel.fileInterface.esterel.EsterelParser
    */
    public LocalSignalDeclarationEsterelStatement(
    final EsterelParser theparser)
        throws EsterelParserException {
        super(theparser);
        this.aStatement = -1;
        this.parseEsterelStatement(theparser);
    } //public LocalSignalDeclarationEsterelStatement
    //(final EsterelParser theparser) {
    /**
     *Implements the methode from <code>EsterelStatement</code>.
     *  Returns a state representation of the esterel statement
     * @param anEsterelModule the esterel module which
     * becomes a KIEL Statechart
     *@param isFinalState is true if the new state has to be
     * a final state
     * @param theLocalEvents Stack with the local events
     * @param theLocalVariables Stack with all local variables
     * @param theTreesTrapSignals
     *                  Stack with all tree trap signals.
     * @throws Esterel2EstudioException
     *                  if the convertion to Kiel is not working. <br>
     *                  Sideeffects: none
     * @return a State
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     * @see kiel.dataStructure.State
     */
    public final kiel.dataStructure.State convertToKiel(
           final EsterelModule anEsterelModule,
           final boolean isFinalState,
           final ArrayList theLocalEvents,
           final ArrayList theLocalVariables,
           final ArrayList theTreesTrapSignals)
           throws Esterel2EstudioException {
        InitialArc anInitialArc = new InitialArc();
        InitialState anInitialState = new InitialState();
        CompositeState aState = null;
        if (isFinalState) {
            aState =
                new FinalORState(this.getEsterelStatementName() + "state");
        } else {
            aState = new ORState(this.getEsterelStatementName() + "state");
        }
        Event[] theEvents = new Event[this.theLocalSignals.length];
        for (int i = 0; i < this.theLocalSignals.length; i++) {
            theEvents[i] =
```

```
                this.theLocalSignals[i].convertToKiel(anEsterelModule);
        }
        StateChartHelpers.addLocalEvents(aState, theEvents);
        int oldsize = theLocalEvents.size();
        theLocalEvents.addAll(aState.getLocalEvents());
        State aStatementState =
            (
                (EsterelStatement) anEsterelModule.getEsterelProgram().get(
                    this.aStatement)).convertToKiel(
                anEsterelModule,
                true,
                theLocalEvents,
                theLocalVariables,
                theTreesTrapSignals);
        //true, because we have only one state
        // we added all the localsignals to theLocalEvenets
        // now we delete them again
        while (theLocalEvents.size() != oldsize) {
            theLocalEvents.remove(theLocalEvents.size() - 1);
        }
        anInitialArc.setSource(anInitialState);
        anInitialArc.setTarget(aStatementState);
        anInitialState.setParent(aState);
        aStatementState.setParent(aState);
        aState.addSubnode(aStatementState);
        aState.addSubnode(anInitialState);
        return aState;
    } //public final kiel.dataStructure.State convertToKiel(
    /**
     * Implemtents the abstrac methode from <code>EsterelStatement</code>.
     * Parses an esterel statment in an <code>org.jdom.Document</code>
     * representation of an .exp file. if you have not called the
     * constructor <code> EsterelStatement(EsterelParser theparser)</code>
     * use the methode <code>preParseEsterelStatement</code> before
     * <code>parseEsterelStatement</code><br>
     * Sideeffects: the class variables are set Changes
     * <code>EsterelParser</code> class variables
     *
     * @param theparser
     *                an instance of <code>EsterelParser</code>
     * @throws EsterelParserException
     *                is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     * @see EsterelStatement#EsterelStatement(EsterelParser theparser)
     * @see #parseEsterelStatement(EsterelParser)
     * @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see kiel.fileInterface.esterel.EsterelParser
     * @see org.jdom.Document
     */
    public final void parseEsterelStatement(
            final EsterelParser theparser)
            throws EsterelParserException {
        String theSymbolTable = "";
        ArrayList aSignalDeclarationList = new ArrayList();
        List anElementList =
            DOMHelpers.getElements(theparser.getXMLDocument(),
                this.getXPathSearchString());
        //get all children of signal
        if (anElementList.size() == 2) {
            //Do we have found two children
            theparser.incStatementCounter(); // we have
                                                    // found a
                                                    // statement
            //find the symbol table for this signal
            if (((Element) anElementList.get(1)).getName()
                    .compareTo("SymbolTable") == 0) {
                theSymbolTable = this.getXPathSearchString()
                        + "[local-name(.)='"
                        + ((Element) anElementList.get(1)).getName()
                        + "'][@id='"
                        + ((Element) anElementList.get(1))
                        .getAttributeValue("id")
                        + "']/*";
            } else if (((Element) anElementList.get(1)).getName()
                    .compareTo("Ref") == 0) {
                theSymbolTable = "//*[@id='"
                        + ((Element) anElementList.get(1))
                        .getAttributeValue("id")
                        + "']/*";
            } else {
                throw new EsterelParserException(
                        this.getEsterelStatementName()
                                + " statements incorrect", theparser);
            }
            List theSymbolTableElementList =
                DOMHelpers.getElements(theparser.getXMLDocument(),
                theSymbolTable);
            // create every local signal
            // the first element in the List is useless for our purpose
            for (int j = 1; j < theSymbolTableElementList.size(); j++) {
                Element aSignalElement = ((Element)
                        theSymbolTableElementList.get(j));
                if (aSignalElement.getName().compareTo("Ref") == 0) {
                    // search for the referenz ref
                    aSignalDeclarationList.add(theparser
                        .getEsterelModule(theparser.getactualEsterModule())
                            .getEsterelSignalByID(aSignalElement
                                .getAttributeValue("id")));
                } else if (aSignalElement.getName()
                        .compareTo("SignalSymbol") == 0) {
                    aSignalDeclarationList
                    .add(theparser
                        .getEsterelModule(
                            theparser.getactualEsterModule())
                            .getEsterelSignalByID(
                                aSignalElement.getAttributeValue("id")));
                } else {
                    throw new EsterelParserException(
                            this.getEsterelStatementName()
                                    + " statements incorrect", theparser);
                }
                if (aSignalDeclarationList
```

```
        .get(aSignalDeclarationList.size() - 1) == null) {
      throw new EsterelParserException(
          this.getEsterelStatementName()
              + ": Found no Signal");
      }
    } //for
    aSignalDeclarationList.trimToSize();
    this.theLocalSignals =
        (EsterelSignal[]) aSignalDeclarationList
            .toArray(new EsterelSignal[aSignalDeclarationList.size()]);
    } //if size=2
  } //public void parseEsterelStatement(final EsterelParser theparser)
  /**
   * Sets the index number from <code>EsterelModule</code> of the
   * substatement. <br>
   * Sideeffects: Chages class variable <code>aStatement</code>
   *
   * @param aStatementIndex
   *          the index in an esterelModule
   * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
   */
  public final void setSignalStatement(final int aStatementIndex) {
    this.aStatement = aStatementIndex;
  } //public final void setSignalStatement(final int aStatement)
  /**
   * Fills the substatements in the esterel module.
   *<br> Sideeffects: none
   * @param theparser the actual parser
   * @throws EsterelParserException is only delivered
   * @see kiel.fileInterface.esterel.EsterelParserException
   *  @see kiel.fileInterface.esterel.EsterelParser
   */
  public final void setSubStatments(final EsterelParser theparser)
      throws EsterelParserException {
    int i = 0; // the index of the substatement in an ast file
    List anElementList =
      DOMHelpers.getElements(
          theparser.getXMLDocument(),
          this.getXPathSearchString());
      this.setSignalStatement(theparser.getAddAt());
      theparser.getEsterelModule(
          theparser.getactualEsterModule())
          .addStatementToModule(
          this.aStatement,
          theparser.createStatement(((Element) (anElementList.get(i))))));
    }
    /**
     *Implementents the abstrac methode from <code>EsterelStatement</code>.
     * Returns a string representation of an esterel statement
     * <br>Sideeffects:
     * none
     * @param anEsterelModule an esterel modul representation
     * @return a String representation of an esterel statment
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
     *@see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    public final String toString(final EsterelModule anEsterelModule) {
      String aSignalList = "";
      for (int i = 0; i < this.theLocalSignals.length; i++) {
        aSignalList
            += this.theLocalSignals[i].toString(anEsterelModule);
        aSignalList += ", ";
      }
      return this.getEsterelStatementName()
        + " "
        + aSignalList
        + " in\n"
        + (
          (EsterelStatement) anEsterelModule.getEsterelProgram().get(
              this.aStatement)).toString(
          anEsterelModule)
        + "end "
        + this.getEsterelStatementName()
        + "\n";
    } //public final String toString(final EsterelModule anEsterelModule)
}
```

C.2.31. LocalVariableEsterelStatement

Auflistung C.36: Die Klasse `LocalVariableEsterelStatement`

```
package kiel.fileInterface.esterel.esterel2estudio;
import java.util.ArrayList;
import java.util.List;
import kiel.dataStructure.CompositeState;
import kiel.dataStructure.FinalORState;
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.ORState;
import kiel.dataStructure.State;
import kiel.dataStructure.Variable;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StateChartHelpers;
import org.jdom.Element;
/**
 * <p> This is a subclass of <code>EsterelStatement</code>
 * classes implements the var in statement.</p>
 * <p>Copyright: Copyright (c) 2005</p>
 * <p>Company: Uni Kiel</p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
 * @version $Revision: 1.40 $ last modified $Date: 2006/02/06 18:35:29 $
 * <br>
 */
public class LocalVariableEsterelStatement extends EsterelStatement {
    /** The index number of the substatement in an
     * <code>EsterelModule</code>.
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    private int aStatement;
    /** the local Variables. */
    private EsterelVariable[] myVariables = null;
    /**
    *Simple constructor.
    */
    public LocalVariableEsterelStatement() {
        super();
        this.aStatement = -1;
} //public LocalVariableDeclariationEsterelStatement() {
    /**
    * Parses a Document and set the class varaibles.
    * <br>It calls also the super constuctor<code>EsterelStatement
    * (EsterelParser theparser)</code> .
    * <br>Calls the <code>parseEsterelStatement</code> methode
    * <br>Sideeffects:
    * Changes <code>EsterelParser</code>class variables
    * @param theparser an intace of a EsterelParser
    * @throws EsterelParserException is only delivered
    * @see kiel.fileInterface.esterel.EsterelParserException
    * @see #parseEsterelStatement(EsterelParser)
    * @see EsterelStatement#EsterelStatement(EsterelParser)
    * @see kiel.fileInterface.esterel.EsterelParser
    * @since 1.7
    */
    public LocalVariableEsterelStatement(final EsterelParser theparser)
        throws EsterelParserException {
        super(theparser);
        this.aStatement = -1;
        this.parseEsterelStatement(theparser);
    } //public LocalVariableDeclariationEsterelStatement
    //(final EsterelParser theparser) {
    /**
     *Implements the methode from <code>EsterelStatement</code>.
    *  Returns a state representation of the esterel statement
    * @param anEsterelModule the esterel module which
    * becomes a KIEL Statechart
    *@param isFinalState is true if the new state has to be
     * a final state
     * @param theLocalEvents Stack with the local events
     * @param theLocalVariables Stack with all local variables
     * @param theTreesTrapSignals
     *                 Stack with all tree trap signals.
     * @throws Esterel2EstudioException
     *                 if the convertion to Kiel is not working. <br>
     *                 Sideeffects: none
     * @return a State
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     * @see kiel.dataStructure.State
     */
    public final kiel.dataStructure.State convertToKiel(
           final EsterelModule anEsterelModule,
           final boolean isFinalState,
           final ArrayList theLocalEvents,
           final ArrayList theLocalVariables,
           final ArrayList theTreesTrapSignals)
           throws Esterel2EstudioException {
       InitialArc anInitialArc = new InitialArc();
       InitialState anInitialState = new InitialState();
       CompositeState aState = null;
       if (isFinalState) {
           aState =
               new FinalORState(this.getEsterelStatementName() + "state");
       } else {
           aState = new ORState(this.getEsterelStatementName() + "state");
       }
       Variable[] theVariables =
           new Variable[this.myVariables.length];
       for (int i = 0; i < this.myVariables.length; i++) {
           theVariables[i] =
                   this.myVariables[i].convertToKiel(
                   anEsterelModule,
```

```
                theLocalEvents,
                theLocalVariables);
        }
        StateChartHelpers.addLocalVars(aState, theVariables);
        int oldsize = theLocalVariables.size();
        theLocalVariables.addAll(aState.getVariables());
        State aStatementState =
            (
                (EsterelStatement) anEsterelModule.getEsterelProgram().get(
                    this.aStatement)).convertToKiel(
                anEsterelModule,
                true,
                theLocalEvents,
                theLocalVariables,
                theTreesTrapSignals);
        //true, because we have only one state
        // we added all the localVariables to theLocalEvenets
        // now we delete them again
        while (theLocalVariables.size() != oldsize) {
            theLocalVariables.remove(theLocalVariables.size() - 1);
        }
        anInitialArc.setSource(anInitialState);
        anInitialArc.setTarget(aStatementState);
        anInitialState.setParent(aState);
        aStatementState.setParent(aState);
        aState.addSubnode(aStatementState);
        aState.addSubnode(anInitialState);
        return aState;
    } //public final kiel.dataStructure.State convertToKiel(
    /**
    *Implemtents the abstrac methode from <code>EsterelStatement</code>.
    * Parses an esterel statment in an <code>org.jdom.Document</code>
    * representation of an .exp file.
    *  if you have not called the constructor
    * <code> EsterelStatement(EsterelParser theparser)</code>
    * use the methode <code>preParseEsterelStatement</code>
    * before <code>parseEsterelStatement</code>
    *
     *  <br>Sideeffects:
    * the class variables are set
    * Changes <code>EsterelParser</code>class variables
    * @param theparser an instance of <code>EsterelParser</code>
    * @throws EsterelParserException is only delivered
    * @see kiel.fileInterface.esterel.EsterelParserException
    * @see EsterelStatement#EsterelStatement(EsterelParser theparser)
    * @see #parseEsterelStatement(EsterelParser)
    * @see EsterelStatement#parseEsterelStatement(EsterelParser)
    * @see kiel.fileInterface.esterel.EsterelParser
    * @see org.jdom.Document
    *@since 1.7
    */
    public final void parseEsterelStatement(final EsterelParser theparser)
        throws EsterelParserException {
        ArrayList aVariableDeclarationList = new ArrayList();
        String theSymbolTable = "";
        List anElementList =
            DOMHelpers.getElements(
                theparser.getXMLDocument(),
                this.getXPathSearchString());
        //get all children of Variable
        if (anElementList.size() == 2) {
            //Do we have found two children
            theparser.incStatementCounter(); // we have found a statement
            //find the symbol table for this Variable
            if (((Element) anElementList.get(1))
                .getName()
                .compareTo("SymbolTable")
                == 0) {
                theSymbolTable =
                    this.getXPathSearchString()
                        + "[local-name(.)='"
                        + ((Element) anElementList.get(1)).getName()
                        + "'][@id='"
                        + ((Element) anElementList.get(1)).getAttributeValue(
                            "id")
                        + "']/*";
            } else if (
                ((Element) anElementList.get(1)).getName().compareTo("Ref")
                    == 0) {
                theSymbolTable =
                    "//*[@id='"
                        + ((Element) anElementList.get(1)).getAttributeValue(
                            "id")
                        + "']/*";
            } else {
                throw new EsterelParserException(
                    this.getEsterelStatementName() + " statements incorrect",
                    theparser);
            }
            List theSymbolTableElementList =
                DOMHelpers.getElements(theparser.getXMLDocument(),
                        theSymbolTable);
            // create every local Variable
            // the first element in the List is useless for our purpose
            for (int j = 1; j < theSymbolTableElementList.size(); j++) {
                Element aVariableElement =
                    ((Element) theSymbolTableElementList.get(j));
                if (aVariableElement.getName().compareTo("Ref") == 0) {
                    // search for the referenz ref
                    aVariableDeclarationList.add(
                        new EsterelVariable(
                            theparser,
                            "//*[@id='"
                                + aVariableElement.getAttributeValue("id")
                                + "']/*"));
                } else if (
                    aVariableElement.getName().compareTo("VariableSymbol")
                        == 0) {
                  aVariableDeclarationList.add(
                    new EsterelVariable(
                        theparser,
                        theSymbolTable
```

```
                        + "[local-name(.)='VariableSymbol'][@id='"
                        + aVariableElement.getAttributeValue("id")
                        + "']/*"));
            } else {
                throw new EsterelParserException(
                    this.getEsterelStatementName()
                        + " statements incorrect",
                    theparser);
            }
        } //for
        aVariableDeclarationList.trimToSize();
        this.myVariables =
            (EsterelVariable[]) aVariableDeclarationList
            .toArray(new EsterelVariable[aVariableDeclarationList.size()]);
    } //if size=2
} //public void parseEsterelStatement(final EsterelParser theparser)
/**
 * Fills the substatements in the esterel module.
 *<br> Sideeffects: none
 * @param theparser the actual parser
* @throws EsterelParserException is only delivered
* @see kiel.fileInterface.esterel.EsterelParserException
 *  @see kiel.fileInterface.esterel.EsterelParser
 *
 */
public final void setSubStatments(final EsterelParser theparser)
    throws EsterelParserException {
    int i = 0; // the index of the substatement in an ast file
    List anElementList =
        DOMHelpers.getElements(
            theparser.getXMLDocument(),
            this.getXPathSearchString());
    this.setVariableStatement(theparser.getAddAt());
    theparser.getEsterelModule(theparser.getactualEsterModule())
    .addStatementToModule(
        this.aStatement,
        theparser.createStatement(((Element) (anElementList.get(i)))));
}
/**
 * Sets the index number from <code>EsterelModule</code>
 * of the substatement.
 *  <br>Sideeffects:
 * Chages class variable <code>aStatement</code>
 * @param aStatementindex the index in an esterelModule
 * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
 *
 */
public final void setVariableStatement(final int aStatementindex) {
    this.aStatement = aStatementindex;
} //public final void setVariableStatement(final int aStatement)
/**
 *Implemtents the abstrac methode from <code>EsterelStatement</code>.
* Returns a string representation of an esterel statement
* <br>Sideeffects:
* none
* @param anEsterelModule an esterel modul representation
* @return a String representation of an esterel statment
* @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
*@see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
*
*/
public final String toString(final EsterelModule anEsterelModule) {
    String aVariableList = "";
    for (int i = 0; i < this.myVariables.length; i++) {
        aVariableList
            += this.myVariables[i].toString(
                anEsterelModule);
        aVariableList += ", ";
    }
    return this.getEsterelStatementName()
        + " "
        + aVariableList
        + " in\n"
        + (
            (EsterelStatement) anEsterelModule.getEsterelProgram().get(
                this.aStatement)).toString(
            anEsterelModule)
        + "end "
        + this.getEsterelStatementName()
        + "\n";
} //public final String toString(final EsterelModule anEsterelModule)
}
```

C.2.32. LoopEachEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.32 dargestellt.

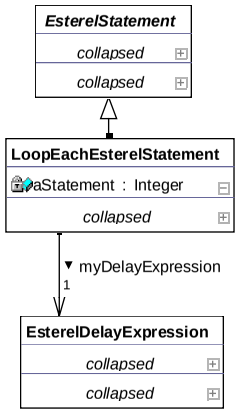


Abbildung C.25.: Klassediagramm `LoopEachEsterelStatement`

Auflistung C.37: Die Klasse `LoopEachEsterelStatement`

```
package kiel.fileInterface.esterel.esterel2estudio;
import java.util.ArrayList;
import java.util.List;
//
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.Node;
import kiel.dataStructure.State;
import kiel.dataStructure.Transition;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StateChartHelpers;
//
import org.jdom.Element;
/**
 * <p> This is a subclass of <code>EsterelStatement</code>
 * classes implements the await statement.</p>
 * It implements await s,await s do p; await case s do p; await case s
 * <p>Copyright: Copyright (c) 2004</p>
 * <p>Company: Uni Kiel</p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
 * @version $Revision: 1.38 $ last modified $Date: 2006/02/06 18:35:29 $
 * <br>
 */
public class LoopEachEsterelStatement extends EsterelStatement {
    /** The index number of the substatement in an
    * <code>EsterelModule</code>.
    * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
    */
    private int aStatement;
    /** The value of the Expressions.*/
    private EsterelDelayExpression myDelayExpression;
    /**
     * Simple constructor.
     */
    public LoopEachEsterelStatement() {
        super();
```

```
        this.aStatement = -1;
        this.myDelayExpression = null;
    }
    /**
    * Parses a Document and set the class varaibles.
    * <br>It calls also the super constuctor<code>EsterelStatement
    * (EsterelParser theparser)</code> .
    * <br>Calls the <code>parseEsterelStatement</code> methode
    * <br>Sideeffects:
    * Changes <code>EsterelParser</code>class variables
    *@param theparser an intace of a EsterelParser
    * @throws EsterelParserException is only delivered
    * @see kiel.fileInterface.esterel.EsterelParserException
    * @see #parseEsterelStatement(EsterelParser)
    * @see EsterelStatement#EsterelStatement(EsterelParser)
    * @see kiel.fileInterface.esterel.EsterelParser
    */
    public LoopEachEsterelStatement(final EsterelParser theparser)
        throws EsterelParserException {
        super(theparser);
        this.aStatement = -1;
        this.myDelayExpression = null;
        if (theparser != null) {
            this.parseEsterelStatement(theparser);
        } else {
            throw new EsterelParserException("no parser");
        }
    }
    /**
     * Returns a state representation of the esterel statement.
     * @param anEsterelModule the esterel module which
     * becomes a KIEL Statechart
     *@param isFinalState is true if the new state has to be
     * a final state
     * @param theLocalEvents Stack with the local events
     * @param theLocalVariables Stack with all local variables
     * @param theTreesTrapSignals
     *                 Stack with all tree trap signals.
     * @throws Esterel2EstudioException
     *                 if the convertion to Kiel is not working. <br>
     *                 Sideeffects: none
     * @return a State
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     * @see kiel.dataStructure.State
     */
    public final kiel.dataStructure.State convertToKiel(
           final EsterelModule anEsterelModule,
           final boolean isFinalState,
           final ArrayList theLocalEvents,
           final ArrayList theLocalVariables,
           final ArrayList theTreesTrapSignals)
           throws Esterel2EstudioException {
        InitialArc aInitialArc = new InitialArc();
        InitialState anInitialState = new InitialState();
        State theStatementState =
            (
                (EsterelStatement)
                    (
                        anEsterelModule.getEsterelProgram().get(
                            this.aStatement))).convertToKiel(
                anEsterelModule,
                false,
                theLocalEvents,
                theLocalVariables,
                theTreesTrapSignals);
        // false means, substate is not a finalstate beacause its aloop
        Transition aTransition =
            StateChartHelpers.createSA(theStatementState, 1, theStatementState);
        // the returned transition is never used,
        aTransition.setLabel(
            this.myDelayExpression.convertToKiel(
                anEsterelModule,
                theLocalEvents,
                theLocalVariables,
                null));
        aInitialArc.setSource(anInitialState);
        aInitialArc.setTarget(theStatementState);

            return StateChartHelpers.createOR(
                this.getEsterelStatementName() + "state",
                new Node[] {anInitialState, theStatementState });

    }
    /**
     *Implemtents the abstrac methode from <code>EsterelStatement</code>
     * Parses an esterel statment in an <code>org.jdom.Document</code>
     * representation of an .exp file
     *  if you have not called the constructor
     * <code> EsterelStatement(EsterelParser theparser)</code>
     * use the methode <code>preParseEsterelStatement</code>
     * before <code>parseEsterelStatement</code>.
     *  <br>Sideeffects:
     * the class variables are set
     * Changes <code>EsterelParser</code>class variables
     * @param theparser an instance of <code>EsterelParser</code>
    * @throws EsterelParserException is only delivered
    * @see kiel.fileInterface.esterel.EsterelParserException
     * @see EsterelStatement#EsterelStatement(EsterelParser)
     *  @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see kiel.fileInterface.esterel.EsterelParser
     * @see org.jdom.Document
     */
    public final void parseEsterelStatement(final EsterelParser theparser)
        throws EsterelParserException {
        List anElementList = null;
        final int index = 1;
        anElementList =
            DOMHelpers.getElements(
                theparser.getXMLDocument(),
                this.getXPathSearchString());
        if (anElementList.size() == 2) {
```

```
        theparser.incStatementCounter();
        this.myDelayExpression =
            new EsterelDelayExpression(
                theparser,
                "//*[local-name(.)='"
                    + ((Element) anElementList.get(index)).getName()
                    + "'][@id='"
                    + ((Element) anElementList.get(index))
                    .getAttributeValue(
                        "id")
                    + "']/*",
                theparser.getXMLDocument());
        this.myDelayExpression.setImmediate(false);
    } else {
        throw new EsterelParserException(
            this.getEsterelStatementName() + " statement wrong ",
            theparser);
    } //else if size
}
/**
 * Fills the substatements in the esterel module.
 * In this class it is fills the do Statements if there are any
 *<br> Sideeffects: none
 * @param theparser the actual parser
 * @throws EsterelParserException is only delivered
 * @see kiel.fileInterface.esterel.EsterelParserException
 * @see kiel.fileInterface.esterel.EsterelParser
 */
public final void setSubStatments(final EsterelParser theparser)
    throws EsterelParserException {
    int i = 0;
    List anElementList = null;
    anElementList =
        DOMHelpers.getElements(
            theparser.getXMLDocument(),
            this.getXPathSearchString());
    if (anElementList.size() == 2) {
        this.aStatement = theparser.getAddAt();
        theparser.getEsterelModule(
            theparser.getactualEsterModule())
            .addStatementToModule(
            theparser.getAddAt(),
            theparser.createStatement(((Element) (anElementList.get(i)))));
    }
}
/**
 *Implementents the abstrac methode from <code>EsterelStatement</code>.
 * Returns a string representation of an esterel statement.
 * @param anEsterelModule an esterel modul representation
 * @return a String representation of an esterel statment
 * @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
 *@see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
 */
public final String toString(final EsterelModule anEsterelModule) {
    return this.getEsterelStatementName()
        + "\n"
        + (
            (EsterelStatement) anEsterelModule.getEsterelProgram().get(
                this.aStatement)).toString(
            anEsterelModule)
        + "each "
        + this.myDelayExpression.toString(anEsterelModule)
        + "\n";
}
}
```

C.2.33. LoopEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.33 dargestellt.

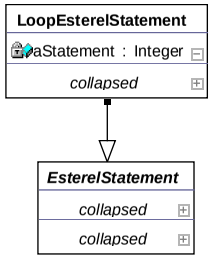


Abbildung C.26.: Klassediagramm `LoopEsterelStatement`

Auflistung C.38: Die Klasse `LoopEsterelStatement`

```
package kiel.fileInterface.esterel.esterel2estudio;
//
import java.util.ArrayList;
import java.util.List;
//
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.Node;
import kiel.dataStructure.State;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StateChartHelpers;
import org.jdom.Element;
/**
 * <p> This is a subclass of <code>EsterelStatement</code>
 * classes implements the loop statement.</p>
 * <p>Copyright: Copyright (c) 2004</p>
 * <p>Company: Uni Kiel</p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
 * @version $Revision: 1.55 $ last modified $Date: 2006/02/06 18:35:29 $
 * <br>
 * $Log: LoopEsterelStatement.java,v $
 * Revision 1.55 2006/02/06 18:35:29 lku
 * *** empty log message ***
 *
 * Revision 1.52 2005/11/10 12:26:26 lku
 * *** empty log message ***
 *
 * Revision 1.44 2005/10/04 11:33:29 lku
 * *** empty log message ***
 *
 * Revision 1.24 2005/05/12 01:37:57 lku
 * *** empty log message ***
 *
 * Revision 1.9 2005/01/14 16:42:26 lku
 * *** empty log message ***
 *
 * <br>
 * Revision 1.8 2005/01/12 15:00:00 lku
 * <br>
 * add javadoc + check for project manual code conventions
 * <br>
 * Revision 1.9 2005/01/14 15:00:00 lku
 * <br>
 * fix javadoc
 * */
public class LoopEsterelStatement extends EsterelStatement {
    /** The index number of the substatement in an
     * <code>EsterelModule</code>.
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    private int aStatement;
```

```
    /**
    *Simple constructor.
    */
    public LoopEsterelStatement() {
        super();
    } //public LoopEsterelStatement() {
    /**
    * Parses a Document and set the class varaibles.
    * <br>It calls also the super constuctor<code>EsterelStatement
    * (EsterelParser theparser)</code> .
    * <br>Calls the <code>parseEsterelStatement</code> methode
    * <br>Sideeffects:
    * Changes <code>EsterelParser</code>class variables
    * @param theparser an intace of a EsterelParser
    * @throws EsterelParserException is only delivered
    * @see kiel.fileInterface.esterel.EsterelParserException
    * @see #parseEsterelStatement(EsterelParser)
    * @see EsterelStatement#EsterelStatement(EsterelParser)
    * @see kiel.fileInterface.esterel.EsterelParser
    */
    public LoopEsterelStatement(final EsterelParser theparser)
        throws EsterelParserException {
        super(theparser);
        this.parseEsterelStatement(theparser);
    } //public LoopEsterelStatement(final EsterelParser theparser) {
    /**
    *Implements the methode from <code>EsterelStatement</code>.
    *  Returns a state representation of the esterel statement
    * @param anEsterelModule the esterel module which
    * becomes a KIEL Statechart
    *@param isFinalState is true if the new state has to be
     * a final state
     * @param theLocalEvents Stack with the local events
    * @param theLocalVariables Stack with all local variables
    * @param theTreesTrapSignals
     *                 Stack with all tree trap signals.
     * @throws Esterel2EstudioException
     *                 if the convertion to Kiel is not working. <br>
     *                 Sideeffects: none
     * @return a State
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     * @see kiel.dataStructure.State
     */
    public final kiel.dataStructure.State convertToKiel(
           final EsterelModule anEsterelModule,
           final boolean isFinalState,
           final ArrayList theLocalEvents,
           final ArrayList theLocalVariables,
           final ArrayList theTreesTrapSignals)
           throws Esterel2EstudioException {
        InitialArc aInitialArc = new InitialArc();
        InitialState anInitialState = new InitialState();
        State theStatementState =
            (
                (EsterelStatement)
                    (
                        anEsterelModule.getEsterelProgram().get(
                            this.aStatement))).convertToKiel(
                anEsterelModule,
                false,
                theLocalEvents,
                theLocalVariables,
                theTreesTrapSignals);
        // false means, substate is not a finalstate beacause its aloop
        StateChartHelpers.createNT(theStatementState, 1, theStatementState);
        // the returned transition is never used,
        aInitialArc.setSource(anInitialState);
        aInitialArc.setTarget(theStatementState);
        // 1 means the priorty

            return StateChartHelpers.createOR(
                this.getEsterelStatementName() + "state",
                new Node[] {anInitialState, theStatementState });

    } //public final kiel.dataStructure.State convertToKiel(
    //final EsterelModule anEsterelModule) {
    /**
    *Implemtents the abstrac methode from <code>EsterelStatement</code>.
    * Parses an esterel statment in an <code>org.jdom.Document</code>
    * representation of an .exp file.
    *  if you have not called the constructor
    * <code> EsterelStatement(EsterelParser theparser)</code>
    * use the methode <code>preParseEsterelStatement</code>
    * before <code>parseEsterelStatement</code>
     *  <br>Sideeffects:
    * the class variables are set
    * Changes <code>EsterelParser</code>class variables
    * @param theparser an instance of <code>EsterelParser</code>
    * @throws EsterelParserException is only delivered
    * @see kiel.fileInterface.esterel.EsterelParserException
    * @see EsterelStatement#EsterelStatement(EsterelParser theparser)
    *  @see EsterelStatement#parseEsterelStatement(EsterelParser)
    * @see kiel.fileInterface.esterel.EsterelParser
    * @see org.jdom.Document
    */
    public final void parseEsterelStatement(final EsterelParser theparser)
        throws EsterelParserException {
        List anElementList = null;
        anElementList =
            DOMHelpers.getElements(
                theparser.getXMLDocument(),
                this.getXPathSearchString());
        if (anElementList.size() == 1) {
            theparser.incStatementCounter();
        } else {
            throw new EsterelParserException(
                this.getEsterelStatementName() + " statement wrong ",
                theparser);
        } //else if size
    } //public final void parseEsterelStatement(final EsterelParser theparser)
    /**
    * Sets the index number from <code>EsterelModule</code>
```

```
     * of the substatement.
     *  <br>Sideeffects:
     * Chages class variable <code>aStatement</code>
     * @param aStatementindex an index of the statement in
     * an EsterelModule
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    public final void setLoopStatement(final int aStatementindex) {
        this.aStatement = aStatementindex;
    } //public final void setLoopStatement(final int aStatement) {
    /**
     * Fills the substatements in the esterel module.
     *<br> Sideeffects: changes EsterelModule
     * @param theparser the actual parser
    * @throws EsterelParserException is only delivered
    * @see kiel.fileInterface.esterel.EsterelParserException
     *  @see kiel.fileInterface.esterel.EsterelParser
     *  @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    public final void setSubStatments(final EsterelParser theparser)
        throws EsterelParserException {
        int i = 0;
        List anElementList = null;
        anElementList =
            DOMHelpers.getElements(
                theparser.getXMLDocument(),
                this.getXPathSearchString());
        if (anElementList.size() == 1) {
            this.setLoopStatement(theparser.getAddAt());
            theparser.getEsterelModule(
                theparser.getactualEsterModule())
                .addStatementToModule(
                theparser.getAddAt(),
                theparser.createStatement(((Element) (anElementList.get(i)))));
        }
    }
    /**
     *Implemtents the abstrac methode from <code>EsterelStatement</code>.
    * Returns a string representation of an esterel statement.
    * <br>Sideeffects:
    * none
    * @param anEsterelModule an esterel modul representation
    * @return a String representation of an esterel statment
    * @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
    *@see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
    */
    public final String toString(final EsterelModule anEsterelModule) {
        return this.getEsterelStatementName()
            + "\n"
            + (
                (EsterelStatement) anEsterelModule.getEsterelProgram().get(
                    this.aStatement)).toString(
                anEsterelModule)
            + "end "
            + this.getEsterelStatementName()
            + "\n";
    } //public final String toString(final EsterelModule anEsterelModule) {
}
```

C.2.34. NothingEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.34 dargestellt.

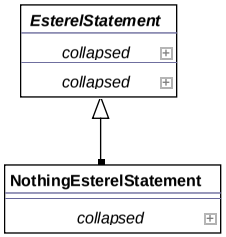


Abbildung C.27.: Klassediagramm `NothingEsterelStatement`

Auflistung C.39: Die Klasse `NothingEsterelStatement`

```
package kiel.fileInterface.esterel.esterel2estudio;
import java.util.ArrayList;
//
import kiel.dataStructure.FinalSimpleState;
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.Node;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.StateChartHelpers;
//
/**
  * <p> This is a subclass of <code>EsterelStatement</code>
 * classes implements the nothing statement.</p>
 * <p>Copyright: Copyright (c) 2004</p>
 * <p>Company: Uni Kiel</p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
 * @version $Revision: 1.53 $ last modified $Date: 2006/02/06 18:35:29 $
 * <br>
 * $Log: NothingEsterelStatement.java,v $
 * Revision 1.53 2006/02/06 18:35:29 lku
 * *** empty log message ***
 *
 * Revision 1.51 2005/11/10 12:26:26 lku
 * *** empty log message ***
 *
 * Revision 1.43 2005/10/04 11:33:29 lku
 * *** empty log message ***
 *
 * Revision 1.23 2005/05/12 01:37:57 lku
 * *** empty log message ***
 *
 * Revision 1.8 2005/01/14 16:42:26 lku
 * *** empty log message ***
 *
 * <br>
 * Revision 1.7 2005/01/12 15:00:00 lku
 * <br>
 * add javadoc + check for project manual code conventions
 * <br>
 * Revision 1.8 2005/01/14 15:00:00 lku
 * <br>
 * fix javadoc
*/
public class NothingEsterelStatement extends EsterelStatement {
    /**
     * Simple constructor.
     */
    public NothingEsterelStatement() {
        super();
    }
    /**
    * Parses a Document and set the class varaibles.
    * <br>It calls also the super constuctor<code>EsterelStatement
    * (EsterelParser theparser)</code> .
    * <br>Calls the <code>parseEsterelStatement</code> methode
```

```
     * <br>Sideeffects:
     * Changes <code>EsterelParser</code>class variables
     * @param theparser an intace of a EsterelParser
     * @see #parseEsterelStatement(EsterelParser)
     * @see EsterelStatement#EsterelStatement(EsterelParser)
     * @see kiel.fileInterface.esterel.EsterelParser
     */
    public NothingEsterelStatement(final EsterelParser theparser) {
        super(theparser);
        this.parseEsterelStatement(theparser);
    } //public NothingEsterelStatement(final EsterelParser theparser) {
    /**
    *Implements the methode from <code>EsterelStatement</code>.
    *  Returns a state representation of the esterel statement
    * @param anEsterelModule the esterel module which
    * becomes a KIEL Statechart
    *@param isFinalState is true if the new state has to be
     * a final state
     * @param theLocalEvents Stack with the local events
    * @param theLocalVariables Stack with all local variables
     * @param theTreesTrapSignals
     *                Stack with all tree trap signals.
     * @throws Esterel2EstudioException
     *                if the convertion to Kiel is not working. <br>
     *                Sideeffects: none
     * @return a State
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     * @see kiel.dataStructure.State
     */
    public final kiel.dataStructure.State convertToKiel(
           final EsterelModule anEsterelModule,
           final boolean isFinalState,
           final ArrayList theLocalEvents,
           final ArrayList theLocalVariables,
           final ArrayList theTreesTrapSignals)
           throws Esterel2EstudioException {
        InitialArc aInitialArc = new InitialArc();
        InitialState anInitialState = new InitialState();
        FinalSimpleState theEndState = new FinalSimpleState();
        aInitialArc.setSource(anInitialState);
        aInitialArc.setTarget(theEndState);
        if (isFinalState) {
            return StateChartHelpers.createFinalOR(
                this.getEsterelStatementName() + "state",
                new Node[] {anInitialState, theEndState });
        }
            return StateChartHelpers.createOR(
                this.getEsterelStatementName() + "state",
                new Node[] {anInitialState, theEndState });

    } //public final kiel.dataStructure.State convertToKiel(
    /**
    *Implemtents the abstrac methode from <code>EsterelStatement</code>
    * Parses an esterel statment in an <code>org.jdom.Document</code>
    * representation of an .exp file
    *  if you have not called the constructor
    * <code> EsterelStatement(EsterelParser theparser)</code>
    * use the methode <code>preParseEsterelStatement</code>
    * before <code>parseEsterelStatement</code>
    * In this special case nothing has to be done.
     *  <br>Sideeffects:
    * none
    * Changes <code>EsterelParser</code>class variables
    * @param theparser an instance of <code>EsterelParser</code>
    * @see EsterelStatement#EsterelStatement(EsterelParser theparser)
    *  @see EsterelStatement#parseEsterelStatement(EsterelParser)
    * @see kiel.fileInterface.esterel.EsterelParser
    * @see org.jdom.Document
    * since 1.7
    */
    public final void parseEsterelStatement(final EsterelParser theparser) { }
    /**
     * Fills the substatements in the esterel module
     * in this class it is empty.
     *<br> Sideeffects: none
     * @param theparser the actual parser
    * @throws EsterelParserException is only delivered
    * @see kiel.fileInterface.esterel.EsterelParserException
     *  @see kiel.fileInterface.esterel.EsterelParser
     * @since 1.9
     */
    public final void setSubStatments(final EsterelParser theparser)
        throws EsterelParserException { }
    /**
     *Implemtents the abstrac methode from <code>EsterelStatement</code>.
    * Returns a string representation of an esterel statement
    * @param anEsterelModule an esterel modul representation
    * @return a String representation of an esterel statment
    * @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
    *@see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
    *@since 1.7
    */
    public final String toString(final EsterelModule anEsterelModule) {
        return this.getEsterelStatementName() + "\n";
    }
}
```

C.2.35. ParallelEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.35 dargestellt.

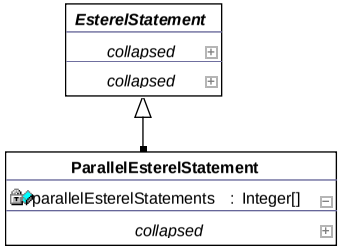


Abbildung C.28.: Klassediagramm `ParallelEsterelStatement`

Auflistung C.40: Die Klasse `ParallelEsterelStatement`

```
package kiel.fileInterface.esterel.esterel2estudio;
//
import java.util.ArrayList;
import java.util.List;
//
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.Node;
import kiel.dataStructure.Region;
import kiel.dataStructure.State;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StateChartHelpers;
//
import org.jdom.Element;
/**
 * <p> This is a subclass of <code>EsterelStatement</code>
 * classes implements the parallel statement.</p>
 * <p>Copyright: Copyright (c) 2004</p>
 * <p>Company: Uni Kiel</p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
 * @version $Revision: 1.55 $ last modified $Date: 2006/02/06 18:35:29 $
 * <br>
 * $Log: ParallelEsterelStatement.java,v $
 * Revision 1.55 2006/02/06 18:35:29 lku
 * *** empty log message ***
 *
 * Revision 1.52 2005/11/10 12:26:26 lku
 * *** empty log message ***
 *
 * Revision 1.44 2005/10/04 11:33:29 lku
 * *** empty log message ***
 *
 * Revision 1.24 2005/05/12 01:37:57 lku
 * *** empty log message ***
 *
 * Revision 1.9 2005/01/14 16:42:26 lku
 * *** empty log message ***
 *
 * <br>
 * Revision 1.8 2005/01/12 15:00:00 lku
 * <br>
 * add javadoc + check for project manual code conventions
 * * <br>
 * Revision 1.8 2005/01/14 15:00:00 lku
 * <br>
 * fix javadoc
 */
public class ParallelEsterelStatement extends EsterelStatement {
    /** The indexes numberes of the substatementes in an
     * <code>EsterelModule</code>.
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    private int[] parallelEsterelStatements;
```

```
    /**
    *Simple constructor.
    */
    public ParallelEsterelStatement() {
        super();
    }
    /**
    * Parses a Document and set the class varaibles.
    * <br>It calls also the super constuctor<code>EsterelStatement
    * (EsterelParser theparser)</code> .
    * <br>Calls the <code>parseEsterelStatement</code> methode
    * <br>Sideeffects:
    * Changes <code>EsterelParser</code>class variables
    * @param theparser an intace of a EsterelParser
    * @see #parseEsterelStatement(EsterelParser)
    * @see EsterelStatement#EsterelStatement(EsterelParser)
    * @see kiel.fileInterface.esterel.EsterelParser
    */
    public ParallelEsterelStatement(final EsterelParser theparser) {
        super(theparser);
        this.parseEsterelStatement(theparser);
    } //public ParallelEsterelStatement(final EsterelParser theparser) {
    /**
    *Implements the methode from <code>EsterelStatement</code>.
    *  Returns a state representation of the esterel statement
    * @param anEsterelModule the esterel module which
    * becomes a KIEL Statechart
    *@param isFinalState is true if the new state has to be
     * a final state
     * @param theLocalEvents Stack with the local events
     * @param theLocalVariables Stack with all local variables
    * @param theTreesTrapSignals
     *                Stack with all tree trap signals.
     * @throws Esterel2EstudioException
     *                if the convertion to Kiel is not working. <br>
     *                Sideeffects: none
     * @return a State
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     * @see kiel.dataStructure.State
     */
    public final kiel.dataStructure.State convertToKiel(
           final EsterelModule anEsterelModule,
           final boolean isFinalState,
           final ArrayList theLocalEvents,
           final ArrayList theLocalVariables,
           final ArrayList theTreesTrapSignals)
           throws Esterel2EstudioException {
        Region[] theRegions = new Region[this.parallelEsterelStatements.length];
        for (int i = 0; i < this.parallelEsterelStatements.length; i++) {
            InitialState anInitialState = new InitialState();
            InitialArc aInitialArc = new InitialArc();
            State aState =
                (
                    (EsterelStatement)
                        (
                            anEsterelModule.getEsterelProgram().get(
                                this
                                    .parallelEsterelStatements[i]))
                                    .convertToKiel(
                            anEsterelModule,
                            true,
                            theLocalEvents,
                            theLocalVariables,
                            theTreesTrapSignals);
            // true means that the substate is final
            aInitialArc.setSource(anInitialState);
            aInitialArc.setTarget(aState);
            theRegions[i] =
                StateChartHelpers.createRegion(
                    new Node[] {anInitialState, aState });
        } //for
        if (isFinalState) {
            return StateChartHelpers.createFinalAND(
                this.getEsterelStatementName() + "state",
                theRegions);
        }
        return StateChartHelpers.createAND(
            this.getEsterelStatementName() + "state",
            theRegions);

    } //public final kiel.dataStructure.State convertToKiel(
    /**
    *Implemtents the abstrac methode from <code>EsterelStatement</code>.
    * Parses an esterel statment in an <code>org.jdom.Document</code>
    * representation of an .exp file.
    *  if you have not called the constructor
    * <code> EsterelStatement(EsterelParser theparser)</code>
    * use the methode <code>preParseEsterelStatement</code>
    * before <code>parseEsterelStatement</code>
     *  <br>Sideeffects:
    * the class variables are set
    * Changes <code>EsterelParser</code>class variables
    * @param theparser an instance of <code>EsterelParser</code>
    * @see EsterelStatement#EsterelStatement(EsterelParser)
    * @see kiel.fileInterface.esterel.EsterelParser
    * @see org.jdom.Document
    */
    public final void parseEsterelStatement(final EsterelParser theparser) {
        List anElementList =
            DOMHelpers.getElements(
                theparser.getXMLDocument(),
                this.getXPathSearchString());
        //stelle fest wieviele Statements noch existieren
        this.parallelEsterelStatements = new int[anElementList.size()];
        // und erhoehe die Anzahl der noch zu Parsenden Statements
        theparser.incStatementCounter(anElementList.size());
    } //public void parseEsterelStatement(EsterelParser theparser)
    /**
    * Sets the index number from <code>EsterelModule</code>
    * of the substatement.
    *  <br>Sideeffects:
    * Chages class variable <code>aStatement</code>
```

```
     * @param aStatementindex an index
     * @param i the index of the <code>EsterelStatement</code> in
     * <code>aStatement</code>
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    public final void setParallelStatement(
        final int aStatementindex,
        final int i) {
      this.parallelEsterelStatements[i] = aStatementindex;
    } //public final void setParallelStatement(final int aStatement, final int
    /**
     * Fills the substatements in the esterel module.
     *<br> Sideeffects: changes EsterelModule
     * @param theparser the actual parser
     * @throws EsterelParserException is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     *  @see kiel.fileInterface.esterel.EsterelParser
     *  @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    public final void setSubStatments(final EsterelParser theparser)
        throws EsterelParserException {
      List anElementList = null;
      anElementList =
          DOMHelpers.getElements(
              theparser.getXMLDocument(),
              this.getXPathSearchString());
      for (int i = 0; i < anElementList.size(); i++) {
        this.setParallelStatement(theparser.getAddAt(), i);
        theparser.getEsterelModule(
            theparser.getactualEsterModule()).addStatementToModule(
            theparser.getAddAt(),
            theparser.createStatement(((Element) (anElementList.get(i)))));
      } // for
    }
```

```
    /**
     *Implemtents the abstrac methode from <code>EsterelStatement</code>.
     * Returns a string representation of an esterel statement
     * <br>Sideeffects:
     * none
     * @param anEsterelModule an esterel modul representation
     * @return a String representation of an esterel statment
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
     *@see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    public final String toString(final EsterelModule anEsterelModule) {
      String giveback = new String("");
      for (int i = 0; i < this.parallelEsterelStatements.length - 1; i++) {
        giveback += "[\n"
            + (
                (EsterelStatement) anEsterelModule.getEsterelProgram().get(
                    this.parallelEsterelStatements[i])).toString(
                anEsterelModule)
            + "]\n||\n";
      }
      return giveback
          + "[\n"
          + (
              (EsterelStatement) anEsterelModule.getEsterelProgram().get(
                  this
                      .parallelEsterelStatements[this
                      .parallelEsterelStatements
                      .length
                      - 1])).toString(
              anEsterelModule)
          + "]\n";
    } //public final String toString(final EsterelModule anEsterelModule) {
}
```

C.2.36. PauseEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.36 dargestellt.

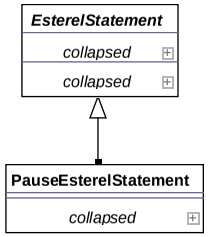


Abbildung C.29.: Klassediagramm `PauseEsterelStatement`

Auflistung C.41: Die Klasse `PauseEsterelStatement`

```
package kiel.fileInterface.esterel.esterel2estudio;
import java.util.ArrayList;

import kiel.dataStructure.FinalSimpleState;
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.Node;
import kiel.dataStructure.SimpleState;
import kiel.dataStructure.Transition;
import kiel.dataStructure.eventexp.Tick;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.StateChartHelpers;
/**
 * <p> This is a subclass of <code>EsterelStatement</code>
 * classes implements pause.</p>
 * <p>Copyright: Copyright (c) 2004</p>
 * <p>Company: Uni Kiel</p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
 * @version $Revision: 1.55 $ last modified $Date: 2006/02/06 18:35:30 $
 * <br>
 * $Log: PauseEsterelStatement.java,v $
 * Revision 1.55 2006/02/06 18:35:30 lku
 * *** empty log message ***
 *
 * Revision 1.53 2005/11/10 12:26:26 lku
 * *** empty log message ***
 *
 * Revision 1.45 2005/10/04 11:33:29 lku
 * *** empty log message ***
 *
 * Revision 1.25 2005/05/12 01:37:57 lku
 * *** empty log message ***
 *
 * Revision 1.10 2005/01/14 16:42:26 lku
 * *** empty log message ***
 *
 * <br>
 * Revision 1.9 2005/01/12 15:00:00 lku
 * <br>
 * add javadoc + check for project manual code conventions
 * Revision 1.10 2005/01/14 15:00:00 lku
 * <br>
 * fix javadoc
*/
public class PauseEsterelStatement extends EsterelStatement {
    /**
     * Simple constructor.
     */
    public PauseEsterelStatement() {
        super();
    }
    /**
     * Parses a Document and set the class varaibles.
     * <br>It calls also the super constuctor<code>EsterelStatement
     * (EsterelParser theparser)</code> .
```

```
     * <br>Calls the <code>parseEsterelStatement</code> methode
     * <br>Sideeffects:
     * Changes <code>EsterelParser</code>class variables
     * @param theparser an intace of a EsterelParser
     * @see #parseEsterelStatement(EsterelParser)
     * @see EsterelStatement#EsterelStatement(EsterelParser)
     * @see kiel.fileInterface.esterel.EsterelParser
     */
    public PauseEsterelStatement(final EsterelParser theparser) {
        super(theparser);
        this.parseEsterelStatement(theparser);
    } //public PauseEsterelStatement(final EsterelParser theparser) {
    /**
    *Implements the methode from <code>EsterelStatement</code>.
    * Returns a state representation of the esterel statement
    * @param anEsterelModule the esterel module which
    * becomes a KIEL Statechart
    *@param isFinalState is true if the new state has to be
     * a final state
     * @param theLocalEvents Stack with the local events
     * @param theLocalVariables Stack with all local variables
     * @param theTreesTrapSignals
     *          Stack with all tree trap signals.
     * @throws Esterel2EstudioException
     *          if the convertion to Kiel is not working. <br>
     *          Sideeffects: none
     *
     * @return a State
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     * @see kiel.dataStructure.State
     */
    public final kiel.dataStructure.State convertToKiel(
            final EsterelModule anEsterelModule,
            final boolean isFinalState,
            final ArrayList theLocalEvents,
            final ArrayList theLocalVariables,
            final ArrayList theTreesTrapSignals)
        throws Esterel2EstudioException {
        int priority = 1;
        InitialArc aInitialArc = new InitialArc();
        InitialState anInitialState = new InitialState();
        SimpleState theStartState = new SimpleState();
        FinalSimpleState theEndState = new FinalSimpleState();
        Transition aWaitTransition =
            StateChartHelpers.createSA(theStartState, priority, theEndState);
        StateChartHelpers.setTrigger(aWaitTransition, new Tick());
        aInitialArc.setSource(anInitialState);
        aInitialArc.setTarget(theStartState);
        if (isFinalState) {
            return StateChartHelpers.createFinalOR(
                    this.getEsterelStatementName() + "state",
                    new Node[] {anInitialState, theStartState, theEndState });
        }
        return StateChartHelpers.createOR(
                this.getEsterelStatementName() + "state",
                new Node[] {anInitialState, theStartState, theEndState });
    }

    //public final kiel.dataStructure.State convertToKiel(
    /**
    *Implements the abstrac methode from <code>EsterelStatement</code>
    * Parses an esterel statment in an <code>org.jdom.Document</code>
    * representation of an .exp file
    * if you have not called the constructor
    * <code> EsterelStatement(EsterelParser theparser)</code>
    * use the methode <code>preParseEsterelStatement</code>
    * before <code>parseEsterelStatement</code>
    * In this special case nothing has to be done.
     * <br>Sideeffects:
    * none
    * Changes <code>EsterelParser</code>class variables
    * @param theparser an instance of <code>EsterelParser</code>
    * @see EsterelStatement#EsterelStatement(EsterelParser)
    * @see kiel.fileInterface.esterel.EsterelParser
    * @see org.jdom.Document
    */
    public final void parseEsterelStatement(final EsterelParser theparser) { }
    /**
     * Fills the substatements in the esterel module
     * in this class it is empty.
     *<br> Sideeffects: none
     * @param theparser the actual parser
    * @throws EsterelParserExeption is only delivered
    * @see kiel.fileInterface.esterel.EsterelParserException
     * @see kiel.fileInterface.esterel.EsterelParser
     */
    public final void setSubStatments(final EsterelParser theparser)
    throws EsterelParserException { }
    /**
     *Implemtents the abstrac methode from <code>EsterelStatement</code>.
    * Returns a string representation of an esterel statement.
    * @param anEsterelModule an esterel modul representation
    * @return a String representation of an esterel statment
    * @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
    *@see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
    */
    public final String toString(final EsterelModule anEsterelModule) {
        return this.getEsterelStatementName();
    } //public final String toString(final EsterelModule aEsterelModule) {
}
```

C.2.37. PresentThenElse

Auflistung C.42: Die Klasse `PresentThenElse`

```
package kiel.fileInterface.esterel.esterel2estudio;
//
import java.util.ArrayList;
import java.util.List;
//
import kiel.dataStructure.ConditionalTransition;
import kiel.dataStructure.DynamicChoice;
import kiel.dataStructure.FinalSimpleState;
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.Node;
import kiel.dataStructure.State;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StateChartHelpers;
//
import org.jdom.Element;
/**
 * <p> This is a subclass of <code>EsterelStatement</code>
 * classes. It parses the esterel statement present and converts it to KIEL.</p>
 * <p>Copyright: Copyright (c) 2004</p>
 * <p>Company: Uni Kiel</p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
 * @version $Revision: 1.56 $ last modified $Date: 2006/02/06 18:35:30 $
 * <br>
 * $Log: PresentThenElse.java,v $
 * Revision 1.56 2006/02/06 18:35:30 lku
 * *** empty log message ***
 *
 * Revision 1.52 2005/11/10 12:26:26 lku
 * *** empty log message ***
 *
 * Revision 1.44 2005/10/04 11:33:29 lku
 * *** empty log message ***
 *
 * Revision 1.24 2005/05/12 01:37:57 lku
 * *** empty log message ***
 *
 * Revision 1.9 2005/01/14 16:42:26 lku
 * *** empty log message ***
 *
 * <br>
 * Revision 1.8 2005/01/12 15:00:00 lku
 * <br>
 * add javadoc + check for project manual code conventions
 * <br>
 * Revision 1.9 2005/01/14 16:00:00 lku
 * <br>
 * fix javadoc
 * Revision 1.9 2005/02/22 16:00:00 lku
 * <br>
 * change Choice to DynamicChioce in converttoKiel
 * 01/03/2005 12:00
 * change aSignalExpression form String to EsterelSignalExpression
*/
public class PresentThenElse extends EsterelStatement {
    /** The SignalExpressions.*/
    private EsterelSignalExpression[] aSignalExpression;
    /** the index number of the else part substatement in an
    * <code>EsterelModule</code>.
    * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
    */
    private int elsePartStatement;
    /** the index numbers of the then part substatements in an
    * <code>EsterelModule</code>.
    * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
    */
    private int[] theStatements;
    /**
    *Simple constructor.
    */
    public PresentThenElse() {
        super(null);
    }
    /**
    * Parses a Document and set the class varaibles.
    * <br>It calls also the super constuctor<code>EsterelStatement
    * (EsterelParser theparser)</code>.
    * <br>Calls the <code>parseEsterelStatement</code> methode
    * <br>Sideeffects:
    * Changes <code>EsterelParser</code>class variables
    * @param theparser an intace of a EsterelParser
    * @throws EsterelParserException is only delivered
    * @see kiel.fileInterface.esterel.EsterelParserException
    * @see #parseEsterelStatement(EsterelParser)
    * @see EsterelStatement#EsterelStatement(EsterelParser)
    * @see kiel.fileInterface.esterel.EsterelParser
    */
    public PresentThenElse(final EsterelParser theparser)
        throws EsterelParserException {
        super(theparser);
        this.theStatements = null;
        this.elsePartStatement = -1;
        this.aSignalExpression = null;
        if (theparser != null) {
            this.parseEsterelStatement(theparser);
        }
    } //public PresentThenElse(final EsterelParser theparser) {
    /**
    *Implements the methode from <code>EsterelStatement</code>.
    *  Returns a state representation of the esterel statement
```

```
     * @param anEsterelModule the esterel module which
     * becomes a KIEL Statechart
     *@param isFinalState is true if the new state has to be
      * a final state
      * @param theLocalEvents Stack with the local events
      * @param theLocalVariables Stack with all local variables
      * @param theTreesTrapSignals
      *                Stack with all tree trap signals.
      * @throws Esterel2EstudioException
      *                if the convertion to Kiel is not working. <br>
      *                Sideeffects: none
      * @return a State
      * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
      * @see kiel.dataStructure.State
      */
     public final kiel.dataStructure.State convertToKiel(
             final EsterelModule anEsterelModule,
             final boolean isFinalState,
             final ArrayList theLocalEvents,
             final ArrayList theLocalVariables,
             final ArrayList theTreesTrapSignals)
             throws Esterel2EstudioException {
         InitialArc aInitialArc = new InitialArc();
         InitialState anInitialState = new InitialState();
         DynamicChoice aPresentTest = new DynamicChoice();
         int nrOfEndStates = this.aSignalExpression.length + 1;
         State[] theEndStates = new State[nrOfEndStates];
         // the cases min 0
         for (int theCaseIndex = 0;
             theCaseIndex < this.aSignalExpression.length;
             theCaseIndex++) {
             if (this.theStatements[theCaseIndex] == -1) {
                 theEndStates[theCaseIndex] = new FinalSimpleState();
             } else {
                 theEndStates[theCaseIndex] =
                     (
                         (EsterelStatement)
                             (
                                 anEsterelModule.getEsterelProgram().get(
                                     this
                                         .theStatements[theCaseIndex])))
                                         .convertToKiel(
                         anEsterelModule,
                         true,
                         theLocalEvents,
                         theLocalVariables,
                         theTreesTrapSignals);
             }
             ConditionalTransition aChoiceTransition =
                 new ConditionalTransition();
             StateChartHelpers.set(
                 aChoiceTransition,
                 aPresentTest,
                 theCaseIndex + 1,
                 theEndStates[theCaseIndex]);
             aChoiceTransition.setLabel(
                     this.aSignalExpression[theCaseIndex].convertToKiel(
                         anEsterelModule,
                         theLocalEvents,
                         theLocalVariables,
                         null));
         } //for
         //the default choice
         if (this.elsePartStatement != -1) {
             theEndStates[nrOfEndStates - 1] =
                 (
                     (EsterelStatement)
                         (
                             anEsterelModule.getEsterelProgram().get(
                                 this.elsePartStatement))).convertToKiel(
                     anEsterelModule,
                     true,
                     theLocalEvents,
                     theLocalVariables,
                     theTreesTrapSignals);
         } else {
             theEndStates[nrOfEndStates - 1] = new FinalSimpleState();
         }
         ConditionalTransition aChoiceTransition = new ConditionalTransition();
         StateChartHelpers.set(
             aChoiceTransition,
             aPresentTest,
             nrOfEndStates,
             theEndStates[nrOfEndStates - 1]);
         aInitialArc.setSource(anInitialState);
         aInitialArc.setTarget(aPresentTest);
         Node[] theNodes = new Node[theEndStates.length + 2];
         for (int i = 0; i < theNodes.length - 2; i++) {
             theNodes[i] = theEndStates[i];
         }
         theNodes[theNodes.length - 2] = anInitialState;
         theNodes[theNodes.length - 1] = aPresentTest;
         if (isFinalState) {
             return StateChartHelpers.createFinalOR(
                 this.getEsterelStatementName() + "state",
                 theNodes);
         }
             return StateChartHelpers.createOR(
                 this.getEsterelStatementName() + "state",
                 theNodes);

     }
     /**
     *Implemtents the abstrac methode from <code>EsterelStatement</code>.
     * Parses an esterel statment in an <code>org.jdom.Document</code>
     * representation of an .exp file.
     *  If you have not called the constructor
     * <code> EsterelStatement(EsterelParser theparser)</code>
     * use the methode <code>preParseEsterelStatement</code>
     * before <code>parseEsterelStatement</code>.
      *  <br>Sideeffects:
     * the class variables are set
```

```
     * Changes <code>EsterelParser</code>class variables
     * @param theparser an instance of <code>EsterelParser</code>
     * @throws EsterelParserException is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     * @see EsterelStatement#EsterelStatement(EsterelParser)
      * @see kiel.fileInterface.esterel.EsterelParser
     * @see org.jdom.Document
     */
    public final void parseEsterelStatement(final EsterelParser theparser)
        throws EsterelParserException {
        List anElementList =
            DOMHelpers.getElements(
                theparser.getXMLDocument(),
                this.getXPathSearchString());
        int numberOfPredicated = 0;
        if (((Element) anElementList.get(anElementList.size() - 1)).getName()
            == "PredicatedStatement") {
            numberOfPredicated = anElementList.size();
        } else {
            numberOfPredicated = anElementList.size() - 1;
            theparser.incStatementCounter();
        }
        this.aSignalExpression =
            new EsterelSignalExpression[numberOfPredicated];
        this.theStatements = new int[numberOfPredicated];
        //set default values
        for (int i = 0; i < numberOfPredicated; i++) {
            this.aSignalExpression[i] = null;
            this.theStatements[i] = -1;
        }
        int aCaseStatementCounter = 0;
        for (int theCaseIndex = 0;
            theCaseIndex < numberOfPredicated;
            theCaseIndex++) {
            if (((Element) anElementList.get(theCaseIndex)).getName()
                == "PredicatedStatement") {
                String aPredicatedStatementXPathSearchString =
                    this.getXPathSearchString()
                        + "[local-name(.)='"
                        + ((Element) anElementList.get(theCaseIndex)).getName()
                        + "'][@id='"
                        + (
                            (Element) anElementList.get(
                                theCaseIndex)).getAttributeValue(
                            "id")
                        + "']/*";
                List aPredicatedElementList =
                    DOMHelpers.getElements(
                        theparser.getXMLDocument(),
                        aPredicatedStatementXPathSearchString);
                if (aPredicatedElementList.size() > 1) {
                    aCaseStatementCounter++;
                }
                int index = aPredicatedElementList.size() - 1;
                // the SignalExpression is the second listelement
                String aSubPredicatedStatementXPathSearchString =
                    "//*[local-name(.)='"
                        + ((Element) aPredicatedElementList.get(index))
                        .getName()
                        + "'][@id='"
                        + (
                            (Element) aPredicatedElementList.get(
                                index)).getAttributeValue(
                            "id")
                        + "']/*";
                this.aSignalExpression[theCaseIndex] =
                    new EsterelSignalExpression(
                        theparser,
                        aSubPredicatedStatementXPathSearchString,
                        theparser.getXMLDocument());
            } else {
                throw new EsterelParserException(
                    this.getEsterelStatementName() + " wrong statement",
                    theparser);
            } //else if size*/
        }
        theparser.incStatementCounter(aCaseStatementCounter);
    } //public final void parseEsterelStatement(final EsterelParser theparser)
    //  final EsterelModule anEsterelModule)
    /**
     * Fills the substatements in the esterel module.
     *<br> Sideeffects: changes EsterelModule
     * @param theparser the actual parser
    * @throws EsterelParserException is only delivered
    * @see kiel.fileInterface.esterel.EsterelParserException
     *  @see kiel.fileInterface.esterel.EsterelParser
     *  @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    public final void setSubStatments(final EsterelParser theparser)
        throws EsterelParserException {
        List anElementList =
            DOMHelpers.getElements(
                theparser.getXMLDocument(),
                this.getXPathSearchString());
        for (int theCaseIndex = 0;
            theCaseIndex < anElementList.size();
            theCaseIndex++) {
            if (((Element) anElementList.get(theCaseIndex)).getName()
                == "PredicatedStatement") {
                String aPredicatedStatementXPathSearchString =
                    this.getXPathSearchString()
                        + "[local-name(.)='"
                        + ((Element) anElementList.get(theCaseIndex)).getName()
                        + "'][@id='"
                        + (
                            (Element) anElementList.get(
                                theCaseIndex)).getAttributeValue(
                            "id")
                        + "']/*";
                List aPredicatedElementList =
                    DOMHelpers.getElements(
                        theparser.getXMLDocument(),
```

```
                    aPredicatedStatementXPathSearchString);
            if (aPredicatedElementList.size() > 1) {
                this.theStatements[theCaseIndex] = theparser.getAddAt();
                theparser.getEsterelModule(
                    theparser.getactualEsterModule()).addStatementToModule(
                    theparser.getAddAt(),
                    theparser.createStatement(
                        ((Element) (aPredicatedElementList.get(0)))));
            } else {
                this.theStatements[theCaseIndex] = -1;
            }
        } else {
            this.elsePartStatement = theparser.getAddAt();
            theparser.getEsterelModule(
                theparser.getactualEsterModule()).addStatementToModule(
                theparser.getAddAt(),
                theparser.createStatement(
                    ((Element) (anElementList
                        .get(anElementList.size() - 1)))));
        }
    } //for
} //set
/**
 *Implemtents the abstrac methode from <code>EsterelStatement</code>.
* Returns a string representation of an esterel statement.
* <br>Sideeffects:
* none
* @param anEsterelModule an esterel modul representation
* @return a String representation of an esterel statment
* @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
*@see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
*/
public final String toString(final EsterelModule anEsterelModule) {
    String result = this.getEsterelStatementName();
    if (this.aSignalExpression.length > 1) {
        for (int theCaseIndex = 0;
            theCaseIndex < this.aSignalExpression.length;
            theCaseIndex++) {
            result += "\n"
                + "     case "
                + this.aSignalExpression[theCaseIndex]
                                .toString(anEsterelModule);
            if (this.theStatements[theCaseIndex] != -1) {
                result += " do "
                    + (
                        (EsterelStatement) anEsterelModule
                            .getEsterelProgram()
                            .get(
                            this.theStatements[theCaseIndex])).toString(
                        anEsterelModule);
            }
        }
    } else {
        result += " " + this.aSignalExpression[0].toString(anEsterelModule);
        if (this.theStatements[0] != -1) {
            result += " then "
                + "\n"
                + (
                    (EsterelStatement) anEsterelModule
                        .getEsterelProgram()
                        .get(
                        this.theStatements[0])).toString(
                    anEsterelModule);
        } else {
            result += "\n";
        }
    }
    if (this.elsePartStatement > -1) {
        result += "else "
            + (
                (EsterelStatement) anEsterelModule.getEsterelProgram().get(
                    this.elsePartStatement)).toString(
                anEsterelModule);
    } //elsePart
    result += "\n end " + this.getEsterelStatementName();
    return result;
  }
} //class
```

C.2.38. ProcedureCallEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.38 dargestellt.

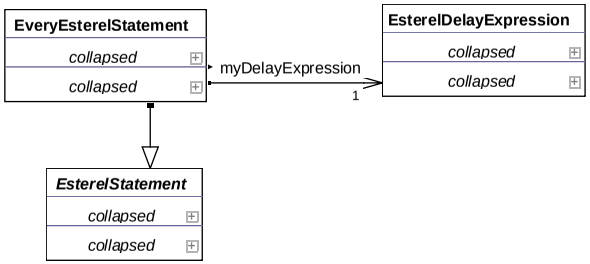


Abbildung C.30.: Klassediagramm `ProcedureCallEsterelStatement`

Auflistung C.43: Die Klasse `ProcedureCallEsterelStatement`

```
package kiel.fileInterface.esterel.esterel2estudio;
import java.util.ArrayList;
import java.util.List;
import kiel.dataStructure.FinalSimpleState;
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.Node;
import kiel.dataStructure.StringLabel;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StateChartHelpers;
import org.jdom.Element;
/**
 * <p> This is a subclass of <code>EsterelStatement</code>
 * classes implements the call statement.</p>
  * <p>Copyright: Copyright (c) 2005</p>
 * <p>Company: Uni Kiel</p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
 * @version $Revision: 1.28 $ last modified $Date: 2006/02/06 18:35:30 $
 * <br>
 */
public class ProcedureCallEsterelStatement extends EsterelStatement {

/**
* the procedure name.
*/
private EsterelProcedureDeclaration myProcedure = null;
/**
 * the reference parameters.
 */
private EsterelVariable[] myReferenceParameter;
/**
 * the value parameters.
 */
private EsterelExpression[] myValueParameter;
/**
 * simple constructor.
 */
public ProcedureCallEsterelStatement() {
    super();
    this.myReferenceParameter = null;
    this.myValueParameter = null;

}
/**
* Parses a Document and set the class varaibles.
* <br>It calls also the super constuctor<code>EsterelStatement
* (EsterelParser theparser)</code> .
* <br>Calls the <code>parseEsterelStatement</code> methode
```

```
     * <br>Sideeffects:
     * Changes <code>EsterelParser</code>class variables
     *@param theparser an intace of a EsterelParser
     * @throws EsterelParserException is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     * @see #parseEsterelStatement(EsterelParser)
     * @see EsterelStatement#EsterelStatement(EsterelParser)
     * @see kiel.fileInterface.esterel.EsterelParser
     */
    public ProcedureCallEsterelStatement(final EsterelParser theparser)
        throws EsterelParserException {
        super(theparser);
        this.myReferenceParameter = null;
        this.myValueParameter = null;
        if (this.getXPathSearchString() != null && theparser != null) {
            this.parseEsterelStatement(theparser);
        } else {
            throw new EsterelParserException(
"ProcedureCall : no parser or path");
        }
    }
    /**
     *Implements the methode from <code>EsterelStatement</code>.
     *  Returns a state representation of the esterel statement
     * @param anEsterelModule the esterel module which
     * becomes a KIEL Statechart
     *@param isFinalState is true if the new state has to be
      * a final state
      * @param theLocalEvents Stack with the local events
     * @param theLocalVariables Stack with all local variables
      * @param theTreesTrapSignals
      *                 Stack with all tree trap signals.
      * @throws Esterel2EstudioException
      *                 if the convertion to Kiel is not working. <br>
      *                 Sideeffects: none
      * @return a State
      * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
      * @see kiel.dataStructure.State
      */
    public final kiel.dataStructure.State convertToKiel(
           final EsterelModule anEsterelModule,
           final boolean isFinalState,
           final ArrayList theLocalEvents,
           final ArrayList theLocalVariables,
           final ArrayList theTreesTrapSignals)
           throws Esterel2EstudioException {
        InitialArc aInitialArc = new InitialArc();
        InitialState anInitialState = new InitialState();
        FinalSimpleState theEndState = new FinalSimpleState();
        aInitialArc.setSource(anInitialState);
        aInitialArc.setTarget(theEndState);
        StringLabel aLabel =
            new StringLabel("/ " + this.toString(anEsterelModule));
        aInitialArc.setLabel(aLabel);
        if (isFinalState) {
            return StateChartHelpers.createFinalOR(
                this.getEsterelStatementName() + "state",
                new Node[] {anInitialState, theEndState });
        }
            return StateChartHelpers.createOR(
                this.getEsterelStatementName() + "state",
                new Node[] {anInitialState, theEndState });

    } //public final kiel.dataStructure.State convertToKiel(
    /**
     * @return Returns the myName.
     */
    public final String getMyName() {
        return this.myProcedure.getProcedureName();
    }
    /**
     *Implemtents the abstrac methode from <code>EsterelStatement</code>
     * Parses an esterel statment in an <code>org.jdom.Document</code>
     * representation of an .exp file
     *  if you have not called the constructor
     * <code> EsterelStatement(EsterelParser theparser)</code>
     * use the methode <code>preParseEsterelStatement</code>
     * before <code>parseEsterelStatement</code>.
      *  <br>Sideeffects:
     * the class variables are set
     * Changes <code>EsterelParser</code>class variables
     * @param theparser an instance of <code>EsterelParser</code>
    * @throws EsterelParserException is only delivered
    * @see kiel.fileInterface.esterel.EsterelParserException
     * @see EsterelStatement#EsterelStatement(EsterelParser)
     *  @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see kiel.fileInterface.esterel.EsterelParser
     * @see org.jdom.Document
     */
    public final void parseEsterelStatement(final EsterelParser theparser)
        throws EsterelParserException {
        List anElementList =
            DOMHelpers.getElements(
                this.getXPathSearchString(),
                theparser.getXMLDocument());
        int theEOVCounter = 0;
        int i = 0;
       ArrayList theReferenceParameter = new ArrayList();
        ArrayList theValueParameter = new ArrayList();
        while (theEOVCounter < 2 && i < anElementList.size()) {
            Element anElement = (Element) anElementList.get(i);
            if (anElement.getName().compareTo("EOV") == 0) {
                theEOVCounter++;
            } else if (i == 0) { // parse name}
                this.myProcedure =
                    theparser
                    .getEsterelModule(theparser.getactualEsterModule())
                    .getEsterelProcedureByID(
                    anElement.getAttributeValue("id"));
            } else if (theEOVCounter == 0) {
                theReferenceParameter.add(
```

```
        theparser
        .getEsterelModule(theparser.getactualEsterModule())
        .getEsterelVariableByID(anElement
                .getAttributeValue("id")));
    } else if (theEOVCounter == 1) {
        theValueParameter.add(
            new EsterelExpression(
                theparser,
                "//*[@id='"
                    + anElement.getAttributeValue("id")
                    + "']/*",
                theparser.getXMLDocument()));
        }
        i++;
    } //while
    this.myReferenceParameter =
        new EsterelVariable[theReferenceParameter.size()];
    for (int j = 0; j < theReferenceParameter.size(); j++) {
        this.myReferenceParameter[j] =
            (EsterelVariable) theReferenceParameter.get(j);
    }
    this.myValueParameter = new EsterelExpression[theValueParameter.size()];
    for (int j = 0; j < theValueParameter.size(); j++) {
        this.myValueParameter[j] =
            (EsterelExpression) theValueParameter.get(j);
    }
}
/**
* Fills the substatements in the esterel module.
* In this class it is fills the do Statements if there are any
*<br> Sideeffects: none
* @param theparser the actual parser
* @throws EsterelParserException is only delivered
* @see kiel.fileInterface.esterel.EsterelParserException
* @see kiel.fileInterface.esterel.EsterelParser
*/
public final void setSubStatments(final EsterelParser theparser)
    throws EsterelParserException { }
/**
*Implemtents the abstrac methode from <code>EsterelStatement</code>.
* Returns a string representation of an esterel statement.
* @param anEsterelModule an esterel modul representation
* @return a String representation of an esterel statment
* @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
*@see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
*/
public final String toString(final EsterelModule anEsterelModule) {
    String result = "call "
        + this.myProcedure.getProcedureName()
        + " ( ";
    if (this.myReferenceParameter != null
        && this.myReferenceParameter.length > 0) {
        for (int i = 0; i < this.myReferenceParameter.length - 1; i++) {
            result
                += this.myReferenceParameter[i].getVariableIdentifier()
                    + Esterel2EstudioProperties.getSeperatorString();
        }
        result += this.myReferenceParameter[
                                this.myReferenceParameter
                                    .length - 1]
                                .getVariableIdentifier();
    }
    result += " ) ( ";
    if (this.myValueParameter != null
        && this.myValueParameter.length > 0) {
        for (int i = 0; i < this.myValueParameter.length - 1; i++) {
            result
                += this.myValueParameter[i]
                .toString(anEsterelModule)
                + Esterel2EstudioProperties.getSeperatorString();
        }
        result += this.myValueParameter[
                            this.myValueParameter.length - 1]
                .toString(anEsterelModule);
    }
    result += " )";
    return result;
}
}
```

C.2.39. RepeatEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.39 dargestellt.

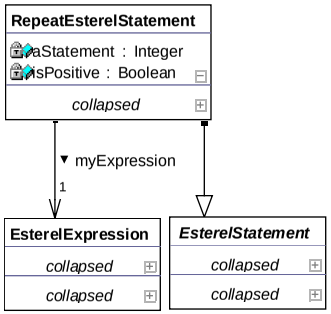


Abbildung C.31.: Klassediagramm `RepeatEsterelStatement`

Auflistung C.44: Die Klasse `RepeatEsterelStatement`

```
package kiel.fileInterface.esterel.esterel2estudio;

import java.util.ArrayList;
import java.util.List;

import kiel.dataStructure.CompositeState;
import kiel.dataStructure.CompoundLabel;
import kiel.dataStructure.ConditionalTransition;
import kiel.dataStructure.DynamicChoice;
import kiel.dataStructure.FinalORState;
import kiel.dataStructure.FinalSimpleState;
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.NormalTermination;
import kiel.dataStructure.ORState;
import kiel.dataStructure.State;
import kiel.dataStructure.StringLabel;
import kiel.dataStructure.TransitionLabel;
import kiel.dataStructure.action.Actions;
import kiel.dataStructure.intexp.IntegerVariable;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StateChartHelpers;

import org.jdom.Element;

/**
 * <p>
 *  This is a subclass of <code>EsterelStatement</code>
 * classes implements the await statement.
 * </p>
 * It implements await s,await s do p; await case s do p; await case s
 * <p>
 * Copyright: Copyright (c) 2004
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.41 $ last modified $Date: 2006/02/17 20:00:22 $
 *              <br>
 */
public class RepeatEsterelStatement
       extends EsterelStatement {
   /**
    * The index number of the substatement in an
```

```
     * <code>EsterelModule</code>.
     *
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    private int aStatement;

    /**
     * true if the repeat is positve.
     */
    private boolean isPositive;

    /**
     * the times of repeat.
     */
    private EsterelExpression myExpression;

    /**
     * Simple constructor.
     */
    public RepeatEsterelStatement() {
        super();
        this.aStatement = -1;
        this.myExpression = null;
        this.isPositive = false;
    }

    /**
     * Parses a Document and set the class varaibles. <br>
     * It calls also the super constuctor <code>EsterelStatement
     * (EsterelParser theparser)</code>.
     * <br>
     * Calls the <code>parseEsterelStatement</code> methode <br>
     * Sideeffects: Changes <code>EsterelParser</code> class variables
     *
     * @param theparser
     *                an intace of a EsterelParser
     * @throws EsterelParserException
     *                 is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     * @see #parseEsterelStatement(EsterelParser)
     * @see EsterelStatement#EsterelStatement(EsterelParser)
     * @see kiel.fileInterface.esterel.EsterelParser
     */
    public RepeatEsterelStatement(final EsterelParser theparser)
            throws EsterelParserException {
        super(
                theparser);
        this.aStatement = -1;
        this.myExpression = null;
        this.isPositive = false;
        this.parseEsterelStatement(theparser);
    }

    /**
     * Returns a state representation of the esterel statement.
     *
     * @param anEsterelModule
     *                the esterel module which becomes a KIEL Statechart
     * @param isFinalState
     *                is true if the new state has to be a final state
     * @param theLocalEvents
     *                Stack with the local events
     * @param theLocalVariables
     *                Stack with all local variables
     * @param theTreesTrapSignals
     *                Stack with all tree trap signals.
     * @throws Esterel2EstudioException
     *                 if the convertion to Kiel is not working. <br>
     *                 Sideeffects: none
     * @return a State
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     * @see kiel.dataStructure.State
     */
    public final kiel.dataStructure.State convertToKiel(
            final EsterelModule anEsterelModule,
            final boolean isFinalState,
            final ArrayList theLocalEvents,
            final ArrayList theLocalVariables,
            final ArrayList theTreesTrapSignals)
            throws Esterel2EstudioException {
        // create two variables
        EsterelExpression oneTimeEx = new EsterelExpression();
        oneTimeEx.setType(anEsterelModule.getEsterelTypeByName("integer"));
        oneTimeEx.setALiteral("0");
        oneTimeEx.setLiteral(true);
        EsterelVariable theTimes = anEsterelModule.addNewVariable();
        theTimes.setExpression(this.myExpression);
        theTimes.setChannelType(this.myExpression.getMyType());
        theTimes.setVariableIdentifier("times");
        EsterelVariable oneTime = anEsterelModule.addNewVariable();
        oneTime.setChannelType(anEsterelModule.getEsterelTypeByName("integer"));
        oneTime.setVariableIdentifier("oneTime");
        oneTime.setExpression(oneTimeEx);
        CompositeState result = null;
        if (isFinalState) {
            result = new FinalORState(
                    this.getEsterelStatementName() + "state");
        } else {
            result = new ORState(
                    this.getEsterelStatementName() + "state");
        }
        result.addVariable((anEsterelModule
                .getEsterelVariableByID(oneTime
                    .getXMLID()))
                    .convertToKiel(anEsterelModule,
            theLocalEvents,
            theLocalVariables));
        result.addVariable((anEsterelModule
                .getEsterelVariableByID(theTimes.getXMLID()))
            .convertToKiel(anEsterelModule,
            theLocalEvents,
            theLocalVariables));
```

```
theLocalVariables.add((anEsterelModule
      .getEsterelVariableByID(oneTime.getXMLID()))
      .convertToKiel(anEsterelModule,
    theLocalEvents,
    theLocalVariables));
theLocalVariables.add((anEsterelModule
      .getEsterelVariableByID(theTimes.getXMLID()))
      .convertToKiel(anEsterelModule,
    theLocalEvents,
    theLocalVariables));
DynamicChoice dc = new DynamicChoice();
FinalSimpleState theEndState = new FinalSimpleState();
InitialState anInitialState = new InitialState();
State theStatementState = ((EsterelStatement)
    (anEsterelModule.getEsterelProgram()
        .get(this.aStatement)))
        .convertToKiel(anEsterelModule,
    false,
    theLocalEvents,
    theLocalVariables,
    theTreesTrapSignals);
// false means, substate is not a finalstate beacause its aloop
IntegerVariable v = (IntegerVariable) (anEsterelModule
      .getEsterelVariableByID(theTimes.getXMLID()))
      .convertToKiel(anEsterelModule,
    theLocalEvents,
    theLocalVariables);
IntegerVariable e = (IntegerVariable) (anEsterelModule
      .getEsterelVariableByID(oneTime.getXMLID()))
      .convertToKiel(anEsterelModule,
    theLocalEvents,
    theLocalVariables);
int priority = 1;
if (!this.isPositive) {
    InitialArc timesRepeatArc = new InitialArc();
    timesRepeatArc.setSource(anInitialState);
    timesRepeatArc.setTarget(theEndState);
    CompoundLabel theCLabel = new CompoundLabel();
    theCLabel.setCondition(StateChartHelpers
          .getBE(anEsterelModule.getEsterelStateChart(),
        theLocalVariables,
        theLocalEvents,
        v.getName() + "<=0"));
    if (theCLabel.getCondition() == null) {
        timesRepeatArc.setLabel(new StringLabel(
              v.getName() + "<=0"));
    }
    timesRepeatArc.setLabel(theCLabel);
    timesRepeatArc.setPriority(priority++);
}
InitialArc defaultArc = new InitialArc();
defaultArc.setSource(anInitialState);
defaultArc.setTarget(theStatementState);
defaultArc.setPriority(2);
NormalTermination theNT = StateChartHelpers.createNT(theStatementState,
    1,
    dc);
TransitionLabel theLabel = null;
Actions theActions =
      StateChartHelpers
          .getActions(anEsterelModule
                .getEsterelStateChart(),
              theLocalVariables,
              theLocalEvents,
            e.getName()
            + ":="
                  + e.getName()
                  + "+1");
if (theActions == null) {
    theLabel = new StringLabel("/"
          + e.getName()
          + ":="
          + e.getName()
          + "+1" );
} else {
    theLabel = new CompoundLabel();
      ((CompoundLabel) theLabel).setEffect(theActions);
}
theNT.setLabel(theLabel);
ConditionalTransition endrepeat = new ConditionalTransition();
ConditionalTransition def = new ConditionalTransition();
endrepeat.setSource(dc);
def.setSource(dc);
endrepeat.setTarget(theEndState);
def.setTarget(theStatementState);
endrepeat.setPriority(1);
def.setPriority(2);
CompoundLabel aCL = new CompoundLabel();
aCL.setCondition(StateChartHelpers
      .getBE(anEsterelModule.getEsterelStateChart(),
    theLocalVariables,
    theLocalEvents,
    v.getName() + "<="
          + e.getName()));
if (aCL.getCondition() == null) {
    endrepeat.setLabel(new StringLabel(
          v.getName() + "<="
                + e.getName()));
} else {
    endrepeat.setLabel(aCL);
}
theLocalVariables.remove(theLocalVariables.size() - 1);
theLocalVariables.remove(theLocalVariables.size() - 1);
result.addSubnode(dc);
result.addSubnode(theStatementState);
result.addSubnode(anInitialState);
result.addSubnode(theEndState);
anInitialState.setParent(result);
theStatementState.setParent(result);
theEndState.setParent(result);
dc.setParent(result);
return result;
```

```
    }

    /**
     * Implemtents the abstrac methode from <code>EsterelStatement</code>
     * Parses an esterel statment in an <code>org.jdom.Document</code>
     * representation of an .exp file if you have not called the
     * constructor <code> EsterelStatement(EsterelParser theparser)</code>
     * use the methode <code>preParseEsterelStatement</code> before
     * <code>parseEsterelStatement</code>.<br>
     * Sideeffects: the class variables are set Changes
     * <code>EsterelParser</code> class variables
     *
     * @param theparser
     *                an instance of <code>EsterelParser</code>
     * @throws EsterelParserException
     *                 is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     * @see EsterelStatement#EsterelStatement(EsterelParser)
     * @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see kiel.fileInterface.esterel.EsterelParser
     * @see org.jdom.Document
     */
    public final void parseEsterelStatement(
            final EsterelParser theparser)
            throws EsterelParserException {
        final int aFOUR = 4;
        List anElementList = null;
        int index = 0;
        anElementList =
            DOMHelpers.getElements(theparser.getXMLDocument(),
            this.getXPathSearchString());
        if (anElementList.size() == aFOUR) {
            theparser.incStatementCounter();
            index++;
            this.myExpression = new EsterelExpression(
                theparser,
                    "//*[local-name(.)='"
                + ((Element) anElementList.get(index))
                .getName()
                          + "'][@id='"
                          + ((Element) anElementList.get(index))
                          .getAttributeValue("id")
                          + "']/*", theparser.getXMLDocument());
            index++;
            if (((Element) anElementList.get(index)).getName()
                    .compareTo("BoolTrue") == 0) {
                this.isPositive = true;
            } else {
                this.isPositive = false;
            }
            index++;
            // dont know what to do at index 3
        } else {
            throw new EsterelParserException(
                    this.getEsterelStatementName() + " statement wrong ",
                            theparser);
        } //else if size
    }

    /**
     * Fills the substatements in the esterel module. In this class it is
     * fills the do Statements if there are any <br>
     * Sideeffects: none
     *
     * @param theparser
     *                 the actual parser
     * @throws EsterelParserException
     *                 is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     * @see kiel.fileInterface.esterel.EsterelParser
     */
    public final void setSubStatments(
            final EsterelParser theparser)
            throws EsterelParserException {
        int i = 0;
        final int aFOUR = 4;
        List anElementList = null;
        anElementList =
            DOMHelpers.getElements(theparser.getXMLDocument(),
            this.getXPathSearchString());
        if (anElementList.size() == aFOUR) {
            this.aStatement = theparser.getAddAt();
            theparser.getEsterelModule(theparser.getactualEsterModule())
                    .addStatementToModule(theparser.getAddAt(),
                        theparser
                        .createStatement(((Element) (anElementList.get(i)))));
        }
    }

    /**
     * Implemtents the abstrac methode from <code>EsterelStatement</code>.
     * Returns a string representation of an esterel statement.
     *
     * @param anEsterelModule
     *                 an esterel modul representation
     * @return a String representation of an esterel statment
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    public final String toString(
            final EsterelModule anEsterelModule) {
        String result = "";
        if (this.isPositive) {
            result += "positive ";
        }
        result += this.getEsterelStatementName() + " "
                + this.myExpression.toString(anEsterelModule)
                + " times \n"
                + ((EsterelStatement) anEsterelModule.getEsterelProgram()
                        .get(this.aStatement)).toString(anEsterelModule)
                + "end "
```

```
    }

    + this.getEsterelStatementName();
    return result + "\n";
  }
```

C.2.40. SequenceEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.40 dargestellt.

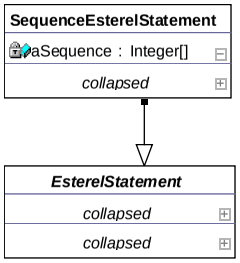


Abbildung C.32.: Klassediagramm `SequenceEsterelStatement`

Auflistung C.45: Die Klasse `SequenceEsterelStatement`

```
package kiel.fileInterface.esterel.esterel2estudio;
//
import java.util.ArrayList;
import java.util.List;
//
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.Node;
import kiel.dataStructure.Transition;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StateChartHelpers;
//
import org.jdom.Element;
/**
 * <p> This is a subclass of <code>EsterelStatement</code>
 * classes implements the sequence statement.</p>
 * <p>Copyright: Copyright (c) 2004</p>
 * <p>Company: Uni Kiel</p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
 * @version $Revision: 1.58 $ last modified $Date: 2006/02/06 18:35:30 $
 * <br>
 */
public class SequenceEsterelStatement extends EsterelStatement {

/** The indexes numberes of the substatementes in an
* <code>EsterelModule</code>.
* @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
*/
private int[] aSequence;
/**
*Simple constructor.
*/
public SequenceEsterelStatement() {
    super();
}
/**
* Parses a Document and set the class varaibles.
* <br>It calls also the super constuctor<code>EsterelStatement
* (EsterelParser theparser)</code> .
* <br>Calls the <code>parseEsterelStatement</code> methode
* <br>Sideeffects:
* Changes <code>EsterelParser</code>class variables
* @param theparser an intace of a EsterelParser
* @see #parseEsterelStatement(EsterelParser)
* @see EsterelStatement#EsterelStatement(EsterelParser)
* @see kiel.fileInterface.esterel.EsterelParser
*/
public SequenceEsterelStatement(final EsterelParser theparser) {
    super(theparser);
    this.parseEsterelStatement(theparser);
} //   public SequenceEsterelStatement(final EsterelParser theparser) {
/**
```

```
*Implements the methode from <code>EsterelStatement</code>.
*  Returns a state representation of the esterel statement
* @param anEsterelModule the esterel module which
* becomes a KIEL Statechart
*@param isFinalState is true if the new state has to be
 * a final state
 * @param theLocalEvents Stack with the local events
 * @param theLocalVariables Stack with all local variables
* @param theTreesTrapSignals
 *                Stack with all tree trap signals.
 * @throws Esterel2EstudioException
 *                if the convertion to Kiel is not working. <br>
 *                Sideeffects: none
 * @return a State
 * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
 * @see kiel.dataStructure.State
 */
public final kiel.dataStructure.State convertToKiel(
       final EsterelModule anEsterelModule,
       final boolean isFinalState,
       final ArrayList theLocalEvents,
       final ArrayList theLocalVariables,
       final ArrayList theTreesTrapSignals)
       throws Esterel2EstudioException {
    Node[] theStates = new Node[this.aSequence.length + 1];
    theStates[theStates.length - 1] = new InitialState();
    InitialArc aInitialArc = new InitialArc();
    for (int i = 0; i < this.aSequence.length; i++) {
        int priority = 1;
        if (i < this.aSequence.length - 1) {
            theStates[i] =
                (
                    (EsterelStatement)
                        (
                            anEsterelModule.getEsterelProgram().get(
                                this.aSequence[i]))).convertToKiel(
                    anEsterelModule,
                    false,
                    theLocalEvents,
                    theLocalVariables,
                    theTreesTrapSignals);
            //false means, that the substate is not final
        } else { // the last state in a sequence has to be final
            theStates[i] =
                (
                    (EsterelStatement)
                        (
                            anEsterelModule.getEsterelProgram().get(
                                this.aSequence[i]))).convertToKiel(
                    anEsterelModule,
                    true,
                    theLocalEvents,
                    theLocalVariables,
                    theTreesTrapSignals);
            //true means, that this substate isfinal
        }
        if (i == 0) {
            aInitialArc.setSource(theStates[theStates.length - 1]);
            aInitialArc.setTarget(theStates[i]);
        } else {
            Object[] allOutTrans =
                (theStates[i - 1].getOutgoingTransitions().toArray());
            for (int iter = 0; iter < allOutTrans.length; iter++) {
                if (allOutTrans[iter] instanceof Transition
                    && ((Transition) allOutTrans[iter])
                        .getPriority()
                        .getValue()
                        >= priority) {
                    priority =
                        ((Transition) allOutTrans[iter])
                            .getPriority()
                            .getValue()
                            + 1;
                }
            }
            StateChartHelpers.createNT(
                theStates[i - 1],
                priority,
                theStates[i]);
        }
    } //for
    if (isFinalState) {
        return StateChartHelpers.createFinalOR(
            this.getEsterelStatementName() + "state",
            theStates);
    }
        return StateChartHelpers.createOR(
            this.getEsterelStatementName() + "state",
            theStates);

} //public final kiel.dataStructure.State convertToKiel(
/**
*Implemtents the abstrac methode from <code>EsterelStatement</code>.
* Parses an esterel statment in an <code>org.jdom.Document</code>
* representation of an .exp file.
*  If you have not called the constructor
* <code> EsterelStatement(EsterelParser theparser)</code>
* use the methode <code>preParseEsterelStatement</code>
* before <code>parseEsterelStatement</code>
 *  <br>Sideeffects:
* the class variables are set
* Changes <code>EsterelParser</code>class variables
* @param theparser an instance of <code>EsterelParser</code>
* @see EsterelStatement#EsterelStatement(EsterelParser)
* @see kiel.fileInterface.esterel.EsterelParser
* @see org.jdom.Document
*/
public final void parseEsterelStatement(final EsterelParser theparser) {
    List anElementList =
        DOMHelpers.getElements(
            theparser.getXMLDocument(),
            this.getXPathSearchString());
```

```
        //get the number of elements
        this.aSequence = new int[anElementList.size()];
        // and increase the number in parser
        theparser.incStatementCounter(anElementList.size());
    } //public void parseEsterelStatement(EsterelParser theparser) {
    /**
     * Sets the index number from <code>EsterelModule</code>
     * of the substatement.
     * <br>Sideeffects:
     * Chages class variable <code>aStatement</code>
     * @param aStatement an index
     * @param i the index of the <code>EsterelStatement</code> in
     * <code>aStatement</code>
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    public final void setSequenceStatement(final int aStatement, final int i) {
        this.aSequence[i] = aStatement;
    } //public final void setSequenceStatement
    /**
     * Fills the substatements in the esterel module.
     *<br> Sideeffects: changes EsterelModule
     * @param theparser the actual parser
     * @throws EsterelParserException is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     *  @see kiel.fileInterface.esterel.EsterelParser
     *  @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    public final void setSubStatments(final EsterelParser theparser)
        throws EsterelParserException {
        List anElementList =
            DOMHelpers.getElements(
                theparser.getXMLDocument(),
                this.getXPathSearchString());
        for (int i = 0; i < anElementList.size(); i++) {
            this.setSequenceStatement(theparser.getAddAt(), i);
            theparser.getEsterelModule(theparser.getactualEsterModule())
                .addStatementToModule(
                    theparser.getAddAt(),
                    theparser.createStatement(((Element) (anElementList.get(i)))));
        } // for
    }
    /**
     *Implementents the abstrac methode from <code>EsterelStatement</code>.
     * Returns a string representation of an esterel statement.
     * <br>Sideeffects:
     * none
     * @param anEsterelModule an esterel modul representation
     * @return a String representation of an esterel statment
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
     *@see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    public final String toString(final EsterelModule anEsterelModule) {
        String giveback = new String("");
        for (int i = 0; i < this.aSequence.length - 1; i++) {
            giveback
                += (
                    (EsterelStatement) anEsterelModule.getEsterelProgram().get(
                        this.aSequence[i])).toString(
                        anEsterelModule)
                + ";\n";
        }
        return giveback
            + (
                (EsterelStatement) anEsterelModule.getEsterelProgram().get(
                    this.aSequence[this.aSequence.length - 1])).toString(
                anEsterelModule)
            + "\n";
    } //public final String toString(final EsterelModule anEsterelModule) {
}
```

C.2.41. SuspendEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.41 dargestellt.

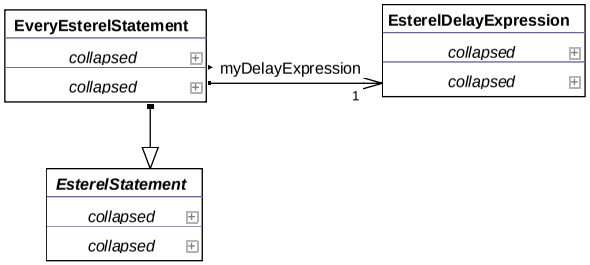


Abbildung C.33.: Klassediagramm `SuspendEsterelStatement`

Auflistung C.46: Die Klasse `SuspendEsterelStatement`

```
package kiel.fileInterface.esterel.esterel2estudio;
//
import java.util.ArrayList;
import java.util.List;
//
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.Node;
import kiel.dataStructure.State;
import kiel.dataStructure.Suspend;
import kiel.dataStructure.Suspension;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StateChartHelpers;
//
import org.jdom.Element;
/**
 * <p> This is a subclass of <code>EsterelStatement</code>
 * classes implements the suspend statement.</p>
 * <p>Copyright: Copyright (c) 2004</p>
 * <p>Company: Uni Kiel</p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
 * @version $Revision: 1.55 $ last modified $Date: 2006/02/06 18:35:30 $
 */
public class SuspendEsterelStatement extends EsterelStatement {
    /** The DelayExpression.*/
    private EsterelDelayExpression aDelayExpression;
    /* The type of the expression*/
    //private String aDelayExpressionType = "";
    /** The index number of the substatement in an
    * <code>EsterelModule</code>.
    * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
    */
    private int aStatement;
    /**
    *Simple constructor.
    */
    public SuspendEsterelStatement() {
        super();
        this.aDelayExpression = null;
    } //public SuspendEsterelStatement() {
    /**
    * Parses a Document and set the class varaibles.
    * <br>It calls also the super constuctor<code>EsterelStatement
    * (EsterelParser theparser)</code> .
    * <br>Calls the <code>parseEsterelStatement</code> methode
    * <br>Sideeffects:
    * Changes <code>EsterelParser</code>class variables
    * @param theparser an intace of a EsterelParser
```

```
     * @throws EsterelParserException is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     * @see #parseEsterelStatement(EsterelParser)
     * @see EsterelStatement#EsterelStatement(EsterelParser)
     * @see kiel.fileInterface.esterel.EsterelParser
     */
    public SuspendEsterelStatement(final EsterelParser theparser)
        throws EsterelParserException {
60      super(theparser);
        this.aStatement = -1;
        this.aDelayExpression = null;
        this.parseEsterelStatement(theparser);
    } //public SuspendEsterelStatement(final EsterelParser theparser) {
    /**
    *Implements the methode from <code>EsterelStatement</code>.
    *  Returns a state representation of the esterel statement.
    * @param anEsterelModule the esterel module which
    * becomes a KIEL Statechart
70  *@param isFinalState is true if the new state has to be
     * a final state
     * @param theLocalEvents Stack with the local events
    * @param theLocalVariables Stack with all local variables
     * @param theTreesTrapSignals
     *                Stack with all tree trap signals.
     * @throws Esterel2EstudioException
     *                if the convertion to Kiel is not working. <br>
     *                Sideeffects: none
     * @return a State
80   * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     * @see kiel.dataStructure.State
     */
    public final kiel.dataStructure.State convertToKiel(
           final EsterelModule anEsterelModule,
           final boolean isFinalState,
           final ArrayList theLocalEvents,
           final ArrayList theLocalVariables,
           final ArrayList theTreesTrapSignals)
           throws Esterel2EstudioException {
90      State theStatementState =
            (
                (EsterelStatement)
                   (
                       anEsterelModule.getEsterelProgram().get(
                          this.aStatement))).convertToKiel(
                anEsterelModule,
                true,
                theLocalEvents,
                theLocalVariables,
100             theTreesTrapSignals);
        // true means, that the substate is final
        Suspend aSuspend = new Suspend();
        Suspension theSuspensionTransition = new Suspension();
        StateChartHelpers.set(
            theSuspensionTransition,
            aSuspend,
            0,
            theStatementState);
        theSuspensionTransition.setLabel(
110         this.aDelayExpression.convertToKiel(
                anEsterelModule,
                theLocalEvents,
                theLocalVariables,
                null));
        InitialState anInitialState = new InitialState();
        InitialArc aInitialArc = new InitialArc();
        aInitialArc.setSource(anInitialState);
        aInitialArc.setTarget(theStatementState);
        if (isFinalState) {
120         return StateChartHelpers.createFinalOR(
                this.getEsterelStatementName() + "state",
                new Node[] {anInitialState, aSuspend, theStatementState });
        }
            return StateChartHelpers.createOR(
                this.getEsterelStatementName() + "state",
                new Node[] {anInitialState, aSuspend, theStatementState });

    }
    /**
130 *Implemtents the abstrac methode from <code>EsterelStatement</code>
    * Parses an esterel statment in an <code>org.jdom.Document</code>
    * representation of an .exp file.
    *  if you have not called the constructor
    * <code> EsterelStatement(EsterelParser theparser)</code>
    * use the methode <code>preParseEsterelStatement</code>
    * before <code>parseEsterelStatement</code>
     *  <br>Sideeffects:
    * the class variables are set
    * Changes <code>EsterelParser</code>class variables
140 * @param theparser an instance of <code>EsterelParser</code>
    * @throws EsterelParserException is only delivered
    * @see kiel.fileInterface.esterel.EsterelParserException
    * @see EsterelStatement#EsterelStatement(EsterelParser)
    * @see kiel.fileInterface.esterel.EsterelParser
    * @see org.jdom.Document
    */
    public final void parseEsterelStatement(final EsterelParser theparser)
        throws EsterelParserException {
        List anElementList =
150         DOMHelpers.getElements(
                theparser.getXMLDocument(),
                this.getXPathSearchString());
        if (anElementList.size() == 2) {
            theparser.incStatementCounter();
            this.aDelayExpression =
                new EsterelDelayExpression(
                    theparser,
                    this.getXPathSearchString()
                        + "[local-name(.)='"
160                     + ((Element) anElementList.get(1)).getName()
                        + "']/*",
                    theparser.getXMLDocument());
        }
```

```
} //public final void parseEsterelStatement(final EsterelParser theparser) {
  /**
   * Fills the substatements in the esterel module
   *<br> Sideeffects: changes EsterelModule.
   * @param theparser the actual parser
   * @throws EsterelParserException is only delivered
   * @see kiel.fileInterface.esterel.EsterelParserException
   *  @see kiel.fileInterface.esterel.EsterelParser
   *  @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
   */
  public final void setSubStatments(final EsterelParser theparser)
      throws EsterelParserException {
      List anElementList =
          DOMHelpers.getElements(
              theparser.getXMLDocument(),
              this.getXPathSearchString());
    this.setSuspendStatement(theparser.getAddAt());
    theparser.getEsterelModule(theparser.getactualEsterModule())
    .addStatementToModule(
        theparser.getAddAt(),
        theparser.createStatement(((Element) (anElementList.get(0)))));
  }
  /**
   * Sets the index number from <code>EsterelModule</code>
   * of the substatement.
   * <br>Sideeffects:
   * Chages class variable <code>aStatement</code>
   * @param aStatementindex an index
   * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
   */
  public final void setSuspendStatement(final int aStatementindex) {
      this.aStatement = aStatementindex;
  }
  /**
   *Implementents the abstrac methode from <code>EsterelStatement</code>
   * Returns a string representation of an esterel statement.
   * <br>Sideeffects:
   * none
   * @param anEsterelModule an esterel modul representation
   * @return a String representation of an esterel statment
   * @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
   *@see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
   */
  public final String toString(final EsterelModule anEsterelModule) {
      return this.getEsterelStatementName()
          + "\n "
          + (
              (EsterelStatement) anEsterelModule.getEsterelProgram().get(
                  this.aStatement)).toString(
              anEsterelModule)
          + "\nwhen "
          + this.aDelayExpression
          + "\n";
  }
}
```

C.2.42. SustainEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.42 dargestellt.

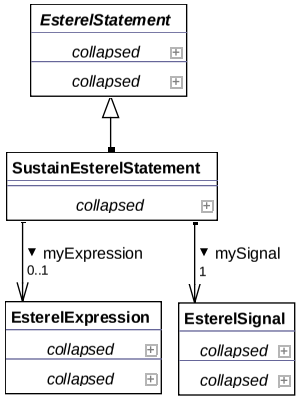


Abbildung C.34.: Klassediagramm `SustainEsterelStatement`

Auflistung C.47: Die Klasse `SustainEsterelStatement`

```
package kiel.fileInterface.esterel.esterel2estudio;

import java.util.ArrayList;
import java.util.List;
import kiel.dataStructure.CompoundLabel;
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.Node;
import kiel.dataStructure.SimpleState;
import kiel.dataStructure.StringLabel;
import kiel.dataStructure.StrongAbortion;
import kiel.dataStructure.TransitionLabel;
import kiel.dataStructure.action.Actions;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.CompoundLabelException;
import kiel.util.CompoundLabelParser;
import kiel.util.DOMHelpers;
import kiel.util.StateChartHelpers;
import org.jdom.Element;

/**
 * <p>
 *  This is a subclass of <code>EsterelStatement</code>
 * classes implements the sustain statement.
 * </p>
 * It implements await s,await s do p; await case s do p; await case s
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.42 $ last modified $Date: 2006/02/06 18:35:30 $
 *               <br>
 */
public class SustainEsterelStatement
        extends EsterelStatement {
    /** The expression of the signal to emit. */
```

```
    private EsterelExpression myExpression;

    /** The signal to emit. */
    private EsterelSignal mySignal = null;

    /**
     * simple constructor.
     */
    public SustainEsterelStatement() {
        super();
        this.myExpression = null;
    }

    /**
     * Parses a Document and set the class varaibles. <br>
     * It calls also the super constuctor <code>EsterelStatement
     * (EsterelParser theparser)</code>.
     * <br>
     * Calls the <code>parseEsterelStatement</code> methode <br>
     * Sideeffects: Changes <code>EsterelParser</code> class variables
     *
     * @param theparser
     *                  an intace of a EsterelParser
     * @throws EsterelParserException
     *                  is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     * @see #parseEsterelStatement(EsterelParser)
     * @see EsterelStatement#EsterelStatement(EsterelParser)
     * @see kiel.fileInterface.esterel.EsterelParser
     */
    public SustainEsterelStatement(final EsterelParser theparser)
           throws EsterelParserException {
        super(
                theparser);
        this.myExpression = null;
        if (theparser != null) {
            this.parseEsterelStatement(theparser);
        } else {
            throw new EsterelParserException(
                    "No Parser");
        }
    } //public SustainEsterelStatement(final EsterelParser theparser)

    /**
     * Returns a state representation of the esterel statement.
     *
     * @param anEsterelModule
     *                  the esterel module which becomes a KIEL Statechart
     * @param isFinalState
     *                  is true if the new state has to be a final state
     * @param theLocalEvents
     *                  Stack with the local events
     * @param theLocalVariables
     *                  Stack with all local variables
     * @param theTreesTrapSignals
     *                  Stack with all tree trap signals.
     * @throws Esterel2EstudioException
     *                  if the convertion to Kiel is not working. <br>
     *                  Sideeffects: none
     * @return a State
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     * @see kiel.dataStructure.State
     */
    public final kiel.dataStructure.State convertToKiel(
           final EsterelModule anEsterelModule,
           final boolean isFinalState,
           final ArrayList theLocalEvents,
           final ArrayList theLocalVariables,
           final ArrayList theTreesTrapSignals)
           throws Esterel2EstudioException {
        InitialArc aInitialArc = new InitialArc();
        InitialState anInitialState = new InitialState();
        SimpleState theStartState = new SimpleState();
        StrongAbortion anEmitTrans = StateChartHelpers.createSA(theStartState,
            1,
            theStartState);
        aInitialArc.setSource(anInitialState);
        aInitialArc.setTarget(theStartState);
        TransitionLabel theLabelofaTransition = null;
        String aLabel = this.mySignal.getSignalIdentifier();
        if (this.myExpression != null) {
            aLabel +=
             "( "
            + this.myExpression.toString(anEsterelModule)
            + ")";
            }
         CompoundLabelParser.getInstance();
         CompoundLabelParser.setStateChart(
                anEsterelModule.getEsterelStateChart());
          CompoundLabelParser.setAllLocals(theLocalVariables, theLocalEvents);
          String aStringLabel = " " + aLabel;
            Actions theActions = null;
            try {
                theActions = CompoundLabelParser.parseActions(aStringLabel);
            } catch (CompoundLabelException ex) {
               theLabelofaTransition = new StringLabel("/" + aLabel);
            }
            if (theLabelofaTransition == null) {
            theLabelofaTransition = new CompoundLabel();
            ((CompoundLabel) theLabelofaTransition).setEffect(theActions);
            }

        aInitialArc.setLabel(theLabelofaTransition);
        anEmitTrans.setLabel(theLabelofaTransition);



        /*
         * It can not be a final state because is not ending. if
         * (isFinalState) { return StateChartHelpers.createFinalOR(
         * this.getEsterelStatementName() + "state", new Node[] {
         * anInitialState, theStartState }); } else {
```

```
         */
        return StateChartHelpers.createOR(this.getEsterelStatementName()
                + "state",
            new Node[] {
                    anInitialState, theStartState });
        //      }
    } //public final kiel.dataStructure.State convertToKiel(

    /**
     * Implemtents the abstrac methode from <code>EsterelStatement</code>
     * Parses an esterel statment in an <code>org.jdom.Document</code>
     * representation of an .exp file if you have not called the
     * constructor <code> EsterelStatement(EsterelParser theparser)</code>
     * use the methode <code>preParseEsterelStatement</code> before
     * <code>parseEsterelStatement</code>.<br>
     * Sideeffects: the class variables are set Changes
     * <code>EsterelParser</code> class variables
     *
     * @param theparser
     *                an instance of <code>EsterelParser</code>
     * @throws EsterelParserException
     *                 is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     * @see EsterelStatement#EsterelStatement(EsterelParser)
     * @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see kiel.fileInterface.esterel.EsterelParser
     * @see org.jdom.Document
     */
    public final void parseEsterelStatement(
            final EsterelParser theparser)
            throws EsterelParserException {
        final int toomuch = 3;
        List anElementList = null;
        anElementList = DOMHelpers.getElements(theparser.getXMLDocument(),
            this.getXPathSearchString());
        if (anElementList.size() > 0
                && anElementList.size() < toomuch) {
            this.mySignal =
                theparser
                .getEsterelModule(theparser.getactualEsterModule())
                .getEsterelSignalByID(this.getSignal(anElementList));
            if (anElementList.size() > 1
                    && anElementList.size() < toomuch) {
                final int theIndex = 1;
                this.myExpression = new EsterelExpression(
                    theparser,
                        this.getXPathSearchString()
                                + "[local-name(.)='"
                                + ((Element) anElementList.get(theIndex))
                                .getName()
                                + "'][@id='"
                                + ((Element) anElementList.get(theIndex))
                                .getAttributeValue("id")
                                + "']/*", theparser.getXMLDocument());
            } //if >1
        } else {
            throw new EsterelParserException(
                    this.getEsterelStatementName()
                            + " no signal expression", theparser);
        } //if >0
    } //public final void parseEsterelStatement(final EsterelParser
      // theparser) {

    /**
     * Fills the substatements in the esterel module. In this class it is
     * fills the do Statements if there are any <br>
     * Sideeffects: none
     *
     * @param theparser
     *                 the actual parser
     * @throws EsterelParserException
     *                 is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     * @see kiel.fileInterface.esterel.EsterelParser
     */
    public final void setSubStatments(
            final EsterelParser theparser)
            throws EsterelParserException {
    }

    /**
     * Implemtents the abstrac methode from <code>EsterelStatement</code>.
     * Returns a string representation of an esterel statement.
     *
     * @param anEsterelModule
     *                 an esterel modul representation
     * @return a String representation of an esterel statment
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    public final String toString(
            final EsterelModule anEsterelModule) {
        return this.getEsterelStatementName()
                + " "
                + this.mySignal.getSignalIdentifier()
                + " (" + this.myExpression
                + ") ";
    }

    /**
     * Retuns the signal of the Signalexpession. <br>
     * Sideeffects: none
     *
     * @param anElementList
     *                 the cildren elements of the SignalExpression
     * @return the signals id
     * @throws EsterelParserException
     *                 if no signal is found
     */
    private String getSignal(
            final List anElementList)
```

```
    throws EsterelParserException {
  final int theSignalIndex = 0;
  if (anElementList.size() > theSignalIndex) {
    return ((Element) anElementList.get(theSignalIndex))
        .getAttributeValue("id");
  }
270
                throw new EsterelParserException(
                        this.getEsterelStatementName()
                                + ": found no signal ");
                }
            }
```

C.2.43. TaskCallEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.43 dargestellt.

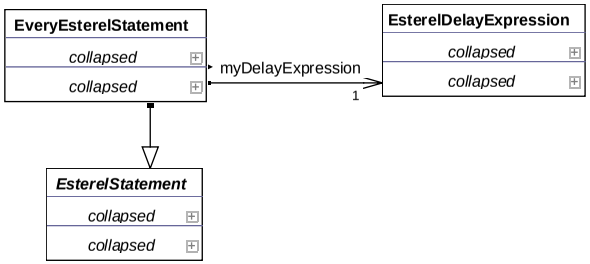


Abbildung C.35.: Klassediagramm `TaskCallEsterelStatement`

Auflistung C.48: Die Klasse `TaskCallEsterelStatement`

```
package kiel.fileInterface.esterel.esterel2estudio;

import java.util.ArrayList;
import java.util.List;
import kiel.dataStructure.FinalSimpleState;
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.Node;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StateChartHelpers;
import org.jdom.Element;

//
/**
 * <p>
 *  This is a subclass of <code>EsterelStatement</code>
 * classes implements the TaskCall statement.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2004
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.36 $ last modified $Date: 2006/02/06 18:35:30 $
 *               <br>
 *               $Log: TaskCallEsterelStatement.java,v $
 *               Revision 1.36 2006/02/06 18:35:30 lku
 *               *** empty log message ***
 *
 */
public class TaskCallEsterelStatement
       extends EsterelStatement {
    /**
     * The reference variable.
     */
    private EsterelVariable[] myReferenceParameter = null;
    /**
     * the substatement.
     */
    private int myStatement;

    /**
     * the name of the task.
     */
    private EsterelTaskDeclaration myTask;
```

```
/**
 * The value parameters.
 */
private EsterelExpression[] myValueParameter = null;

/**
 * the reurn signal.
 */
private EsterelSignal theReturnSignal = null;

/**
 * Simple constructor.
 */
public TaskCallEsterelStatement() {
    super();
    final int m = -2;
    this.myStatement = m;
}

/**
 * Parses a Document and set the class varaibles. <br>
 * It calls also the super constuctor <code>EsterelStatement
 * (EsterelParser theparser)</code>.
 * <br>
 * Calls the <code>parseEsterelStatement</code> methode <br>
 * Sideeffects: Changes <code>EsterelParser</code> class variables
 *
 * @param theparser
 *                an intace of a EsterelParser
 *@throws EsterelParserException aException
 * @see #parseEsterelStatement(EsterelParser)
 * @see EsterelStatement#EsterelStatement(EsterelParser)
 * @see kiel.fileInterface.esterel.EsterelParser
 */
public TaskCallEsterelStatement(final EsterelParser theparser)
       throws EsterelParserException {
    super(
          theparser);
    final int m = -2;
    this.myStatement = m;
    this.parseEsterelStatement(theparser);
} //public TaskCallEsterelStatement(final EsterelParser theparser) {

/**
 * Implements the methode from <code>EsterelStatement</code>.
 * Returns a state representation of the esterel statement
 *
 * @param anEsterelModule
 *                the esterel module which becomes a KIEL Statechart
 * @param isFinalState
 *                is true if the new state has to be a final state
 * @param theLocalEvents
 *                Stack with the local events
 * @param theLocalVariables
 *                Stack with all local variables
 * @param theTreesTrapSignals
 *                Stack with all tree trap signals.
 * @throws Esterel2EstudioException
 *                if the convertion to Kiel is not working. <br>
 *                Sideeffects: none
 * @return a State
 * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
 * @see kiel.dataStructure.State
 */
public final kiel.dataStructure.State convertToKiel(
       final EsterelModule anEsterelModule,
       final boolean isFinalState,
       final ArrayList theLocalEvents,
       final ArrayList theLocalVariables,
       final ArrayList theTreesTrapSignals)
       throws Esterel2EstudioException {
   if (this.myStatement > -1) {
       return ((EsterelStatement) anEsterelModule.getEsterelProgram()
              .get(this.myStatement)).convertToKiel(anEsterelModule,
           true,
           theLocalEvents,
           theLocalVariables,
           theTreesTrapSignals);
   }
   InitialArc aInitialArc = new InitialArc();
   InitialState anInitialState = new InitialState();
   FinalSimpleState theEndState = new FinalSimpleState();
   aInitialArc.setSource(anInitialState);
   aInitialArc.setTarget(theEndState);
   return StateChartHelpers.createFinalOR(this.getEsterelStatementName()
          + "state",
       new Node[] {
              anInitialState, theEndState });
} //public final kiel.dataStructure.State convertToKiel(

/**
 * @return Returns the theReturnSignalID.
 */
public final EsterelSignal getTheReturnSignalID() {
    return this.theReturnSignal;
}

/**
 * Implemtents the abstrac methode from <code>EsterelStatement</code>
 * Parses an esterel statment in an <code>org.jdom.Document</code>
 * representation of an .exp file if you have not called the
 * constructor <code> EsterelStatement(EsterelParser theparser)</code>
 * use the methode <code>preParseEsterelStatement</code> before
 * <code>parseEsterelStatement</code> In this special case TaskCall
 * has to be done. <br>
 * Sideeffects: none Changes <code>EsterelParser</code> class
 * variables
 *
 * @param theparser
 *                an instance of <code>EsterelParser</code>
 *@throws EsterelParserException if something goes wrong.
 * @see EsterelStatement#EsterelStatement(EsterelParser theparser)
```

```
     * @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see kiel.fileInterface.esterel.EsterelParser
     * @see org.jdom.Document
     */
    public final void parseEsterelStatement(
            final EsterelParser theparser)
            throws EsterelParserException {
        final int aTHREE = 3;
        final int aFOUR = 4;
        List anElementList = DOMHelpers.getElements(this.getXPathSearchString(),
            theparser.getXMLDocument());
        int theEOVCounter = 0;
        int i = 0;
        ArrayList theReferenceParameter = new ArrayList();
        ArrayList theValueParameter = new ArrayList();
        while (theEOVCounter < aFOUR
                && i < anElementList.size()) {
            Element anElement = (Element) anElementList.get(i);
            if (anElement.getName().compareTo("EOV") == 0) {
                theEOVCounter++;
            } else if (i == 0) { // parse name}
                this.myTask =
                    theparser
                    .getEsterelModule(theparser.getactualEsterModule())
                    .getEsterelTaskByID(anElement.getAttributeValue("id"));
            } else if (theEOVCounter == 0) {
                theReferenceParameter.add(
                    theparser
                    .getEsterelModule(theparser.getactualEsterModule())
                    .getEsterelVariableByID(anElement.getAttributeValue("id")));
            } else if (theEOVCounter == 1) {
                theValueParameter.add(new EsterelExpression(
                    theparser,
                        "//*[@id='"
                                + anElement.getAttributeValue("id")
                                + "']/*",
                        theparser.getXMLDocument()));
            } else if (theEOVCounter == 2) {
                this.theReturnSignal =
                    theparser
                    .getEsterelModule(theparser.getactualEsterModule())
                    .getEsterelSignalByID(anElement.getAttributeValue("id"));
                theEOVCounter++;
            } else if (theEOVCounter == aTHREE
                    && anElement.getName().compareTo("NULL") != 0) {
                theparser.incStatementCounter();
                this.myStatement++;
            }
            i++;
        } //while}
        this.myReferenceParameter =
            new EsterelVariable[theReferenceParameter.size()];
        for (int j = 0; j < theReferenceParameter.size(); j++) {
            this.myReferenceParameter[j] =
                (EsterelVariable) theReferenceParameter.get(j);
        }
        this.myValueParameter = new EsterelExpression[theValueParameter.size()];
        for (int j = 0; j < theValueParameter.size(); j++) {
            this.myValueParameter[j] =
                (EsterelExpression) theValueParameter.get(j);
        }
    }

    /**
     * Fills the substatements in the esterel module in this class it is
     * empty. <br>
     * Sideeffects: none
     *
     * @param theparser
     *                the actual parser
     * @throws EsterelParserException
     *                is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     * @see kiel.fileInterface.esterel.EsterelParser
     */
    public final void setSubStatments(
            final EsterelParser theparser)
            throws EsterelParserException {
        List anElementList =
            DOMHelpers.getElements(theparser.getXMLDocument(),
            this.getXPathSearchString());
        if (this.myStatement == -1) {
            this.myStatement = theparser.getAddAt();
            theparser.getEsterelModule(theparser.getactualEsterModule())
                    .addStatementToModule(
                            theparser.getAddAt(),
                        theparser.createStatement(
                                ((Element)
                                        (anElementList
                                                .get(anElementList.size()
                                                        - 1)))));
        } // for
    }

    /**
     * Implemtents the abstrac methode from <code>EsterelStatement</code>.
     * Returns a string representation of an esterel statement
     *
     * @param anEsterelModule
     *                an esterel modul representation
     * @return a String representation of an esterel statment
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    public final String toString(
            final EsterelModule anEsterelModule) {
        String result = this.myTask.getTaskName()
                + "(";
        if (this.myReferenceParameter != null
                && this.myReferenceParameter.length > 0) {
            for (int i = 0; i < this.myReferenceParameter.length - 1; i++) {
                result += this.myReferenceParameter[i].getVariableIdentifier()
```

```
            + Esterel2EstudioProperties.getSeperatorString();
    }
    result += this.myReferenceParameter[
                    this.
                    myReferenceParameter
                    .length - 1]
                    .getVariableIdentifier();
  }
  result += ")(";
  if (this.myValueParameter != null
      && this.myValueParameter.length > 0) {
    for (int i = 0; i < this.myValueParameter.length - 1; i++) {
      result += this.myValueParameter[i].toString(anEsterelModule)
            + Esterel2EstudioProperties.getSeperatorString();
    }
    result += this.myValueParameter[
                    this.myValueParameter
                    .length - 1]
                    .toString(anEsterelModule);
  }
  result += ")"
    + " return "
    + this.theReturnSignal.getSignalIdentifier();
  return result;
 }
}
```

C.2.44. TrapEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.44 dargestellt.

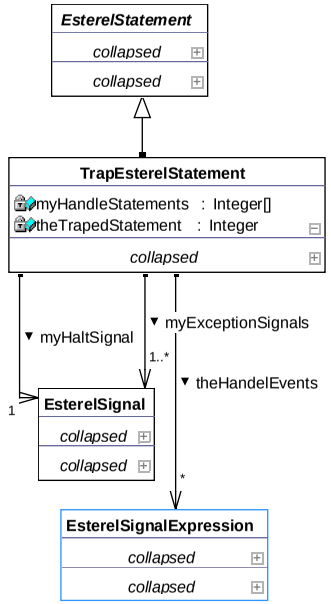


Abbildung C.36.: Klassediagramm `TrapEsterelStatement`

Auflistung C.49: Die Klasse `TrapEsterelStatement`

```
package kiel.fileInterface.esterel.esterel2estudio;

//
import java.util.ArrayList;
import java.util.List;

import kiel.dataStructure.CompositeState;
import kiel.dataStructure.CompoundLabel;
import kiel.dataStructure.ConditionalTransition;
import kiel.dataStructure.DynamicChoice;
import kiel.dataStructure.FinalORState;
import kiel.dataStructure.FinalSimpleState;
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.Node;
import kiel.dataStructure.ORState;
import kiel.dataStructure.Region;
import kiel.dataStructure.SimpleState;
```

```
import kiel.dataStructure.State;
import kiel.dataStructure.StringLabel;
import kiel.dataStructure.Transition;
import kiel.dataStructure.TransitionLabel;
import kiel.dataStructure.WeakAbortion;
import kiel.dataStructure.boolexp.BooleanExpression;
import kiel.dataStructure.boolexp.BooleanOr;
import kiel.dataStructure.eventexp.DelayExpression;
import kiel.dataStructure.eventexp.Event;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StateChartHelpers;

import org.jdom.Element;

/**
 * <p>
 *  This is a subclass of <code>EsterelStatement</code>
 * classes.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2004
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.60 $ last modified $Date: 2006/02/17 20:00:22 $
 *              <br>
 */
public class TrapEsterelStatement
       extends EsterelStatement {
    /** the trap signals. */
    private EsterelSignal[] myExceptionSignals;

    /**
     * the indexes of the handled statements.
     */
    private int[] myHandleStatements;

    /** true if the ExceptionSignal has no handler. */
    private boolean[] pureHandle;

    /**
     * the exception events which starts a handel.
     */
    private EsterelSignalExpression[] theHandleEvents;

    /**
     * the index number of the substatement in an
     * <code>EsterelModule</code>.
     *
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    private int theTrapedStatement;
    /**
    * A new local signal which is emmited when the trap has to hold.
    */
    private EsterelSignal myHaltSignal = null;
    /**
     * Simple constructor.
     */
    public TrapEsterelStatement() {
        super();
        this.theHandleEvents = null;
        this.theTrapedStatement = -1;
        this.myExceptionSignals = null;
        this.myHandleStatements = null;
    }

    /**
     * Parses a Document and set the class varaibles. <br>
     * It calls also the super constuctor <code>EsterelStatement
     * (EsterelParser theparser)</code>.
     * <br>
     * Calls the <code>parseEsterelStatement</code> methode <br>
     * Sideeffects: Changes <code>EsterelParser</code> class variables
     *
     * @param theparser
     *                  an intace of a EsterelParser
     * @throws EsterelParserException
     *                  is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     * @see #parseEsterelStatement(EsterelParser)
     * @see EsterelStatement#EsterelStatement(EsterelParser)
     * @see kiel.fileInterface.esterel.EsterelParser
     */
    public TrapEsterelStatement(final EsterelParser theparser)
           throws EsterelParserException {
        super(
                theparser);
        this.theHandleEvents = null;
        this.theTrapedStatement = -1;
        this.myExceptionSignals = null;
        this.myHandleStatements = null;
        this.parseEsterelStatement(theparser);
        this.myHaltSignal =
            theparser.getEsterelModule(
                theparser.getactualEsterModule())
        .addNewSignal();
        this.myHaltSignal.setTrap(true);
        this.myHaltSignal.setMySignalIdentifier("halt"
               + this.getEsterelStatementName());
        this.myHaltSignal.createNewIdentifierName(
            theparser.getEsterelModule(theparser.getactualEsterModule()));
      } // public TrapEsterelStatement(final EsterelParser theparser) {

    /**
     * Implements the methode from <code>EsterelStatement</code>.
     * Returns a state representation of the esterel statement
```

```
     *
     * @param anEsterelModule
     *          the esterel module which becomes a KIEL Statechart
     * @param isFinalState
     *          is true if the new state has to be a final state
     *
     * @param theLocalEvents
     *          Stack with the local events
     *
     * @param theLocalVariables
     *          Stack with all local variables
     *
     * @param theTreesTrapSignals
     *          Stack with all tree trap signals.
     *
     * @throws Esterel2EstudioException
     *          if the convertion to Kiel is not working. <br>
     *          Sideeffects: none
     *
     * @return a State
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     * @see kiel.dataStructure.State
     */
    public final kiel.dataStructure.State convertToKiel(
            final EsterelModule anEsterelModule,
            final boolean isFinalState,
            final ArrayList theLocalEvents,
            final ArrayList theLocalVariables,
            final ArrayList theTreesTrapSignals)
            throws Esterel2EstudioException {

        int priority = 1;
        InitialArc anInitialArc = new InitialArc();
        InitialState anInitialState = new InitialState();
        CompositeState aState = null;
        FinalSimpleState theNormalTermState = new FinalSimpleState();
        SimpleState theTrapState = new SimpleState();
        State theHandleState = null;
        if (isFinalState) {
            aState = new FinalORState(
                    this.getEsterelStatementName()
                            + "state");

        } else {
            aState = new ORState(
                    this.getEsterelStatementName()
                            + "state");

        }
        // create trap events
        Event[] theEvents = new Event[this.myExceptionSignals.length];
        for (int i = 0; i < this.myExceptionSignals.length; i++) {
            theEvents[i] = this.myExceptionSignals[i]
                    .convertToKiel(anEsterelModule);
        }
        //add them to statechart
        StateChartHelpers.addLocalEvents(aState,
                theEvents);
        int oldsize = theLocalEvents.size();
        theLocalEvents.addAll(aState.getLocalEvents());
        theTreesTrapSignals.add(this);
        State aStatementState = ((EsterelStatement) anEsterelModule
                .getEsterelProgram()
                .get(this.theTrapedStatement))
                .convertToKiel(anEsterelModule,
                        false,
                        theLocalEvents,
                        theLocalVariables,
                        theTreesTrapSignals);
        theTreesTrapSignals.remove(theTreesTrapSignals.size() - 1);
        // if there are this trap has to hold
        WeakAbortion aWA = StateChartHelpers.createWA(aStatementState,
                priority,
                theTrapState);
        priority++;
        String aLabel = this.myHaltSignal.getSignalIdentifier();
        DelayExpression aDE = new DelayExpression();
        aDE.setEventExpression(StateChartHelpers
                .getBE(anEsterelModule.getEsterelStateChart(),
                theLocalVariables,
                theLocalEvents,
                aLabel));
        if (aDE.getEventExpression() == null) {
            aWA.setLabel(new StringLabel("# "
                    + aLabel));
        } else {
            CompoundLabel aCL = new CompoundLabel();
            aDE.setImmediate(true);
            aCL.setTrigger(aDE);
            aWA.setLabel(aCL);
        }
        aState.addSubnode(theTrapState);
        theTrapState.setParent(aState);
        System.out.println(aWA.getLabel().toString());
        //false, there will be a state for normal termination and
        // exception
        // handler used
        if (this.myHandleStatements != null
                && this.myHandleStatements.length > 1) {
            Region[] theRegions = new Region[this.myHandleStatements.length];
            for (int i = 0; i < this.myHandleStatements.length; i++) {
                InitialArc aRegionInitialArc = new InitialArc();
                InitialState aRegionInitialState = new InitialState();
                FinalSimpleState theEndState = new FinalSimpleState();
                DynamicChoice aRegionTest = new DynamicChoice();
                aRegionInitialArc.setSource(aRegionInitialState);
                aRegionInitialArc.setTarget(aRegionTest);
                ConditionalTransition theDefaultTransition =
                    new ConditionalTransition();
                ConditionalTransition aChoiceTransition =
                    new ConditionalTransition();
                // create Handle
                State anExceptionState = ((EsterelStatement)
                        (anEsterelModule
                                .getEsterelProgram()
                                .get(this.myHandleStatements[i])))
                                .convertToKiel(anEsterelModule,
                                        true,
```

```
            theLocalEvents,
            theLocalVariables,
            theTreesTrapSignals);
        StateChartHelpers.set(aChoiceTransition,
            aRegionTest,
            1,
            anExceptionState);
        aChoiceTransition
        .setLabel(
            this.theHandleEvents[i]
                            .convertToKiel(anEsterelModule,
            theLocalEvents,
            theLocalVariables,
            null));
        theDefaultTransition.setPriority(2);
        theDefaultTransition.setSource(aRegionTest);
        theDefaultTransition.setTarget(theEndState);
        theRegions[i] = StateChartHelpers.createRegion(new Node[] {
                aRegionInitialState, aRegionTest, theEndState,
                anExceptionState });
    } //for
    // create the wa to the handles
    theHandleState = StateChartHelpers.createFinalAND("",
        theRegions);
    Transition aTransition = StateChartHelpers.createWA(aStatementState,
        priority++,
        theHandleState);
    // create the lable for the wA
    TransitionLabel theLabel = null;
    for (int i = 0; i < this.theHandleEvents.length; i++) {
        TransitionLabel dummy =
            this.theHandleEvents[i]
                            .convertToKiel(anEsterelModule,
            theLocalEvents,
            theLocalVariables,
            null);
        if (theLabel == null) {
            theLabel = dummy;
        } else if (dummy instanceof CompoundLabel
                && theLabel instanceof CompoundLabel) {
            BooleanExpression aBExp1 =
                ((CompoundLabel) theLabel).getTrigger()
                    .getEventExpression();
            BooleanExpression aBExp2 =
                ((CompoundLabel) dummy).getTrigger()
                    .getEventExpression();
            ((CompoundLabel) theLabel).getTrigger()
                    .setEventExpression(new BooleanOr(
                            aBExp1, aBExp2));
        } else if (dummy instanceof StringLabel
                && theLabel instanceof CompoundLabel) {
            String aBExp1 = ((CompoundLabel) theLabel).toString();
            String aBExp2 = dummy.toString();
            theLabel = new StringLabel(
                    aBExp1
                            + " or " + aBExp2);
        } else if (dummy instanceof CompoundLabel
                && theLabel instanceof StringLabel) {
            String aBExp1 = theLabel.toString();
            String aBExp2 = ((CompoundLabel) dummy).toString();
            theLabel = new StringLabel(
                    aBExp1
                            + " or " + aBExp2);
        } else if (dummy instanceof StringLabel
                && theLabel instanceof StringLabel) {
            String aBExp1 = theLabel.toString();
            String aBExp2 = dummy.toString();
            theLabel = new StringLabel(
                    aBExp1
                            + " or " + aBExp2);
        }
    } // for
    aTransition.setLabel(theLabel);
    StateChartHelpers.setImmediate(aTransition);
} else if (this.myHandleStatements != null
        && this.myHandleStatements.length > 0) {
    theHandleState = ((EsterelStatement)
            (anEsterelModule.getEsterelProgram()
            .get(this.myHandleStatements[0])))
    .convertToKiel(anEsterelModule,
        true,
        theLocalEvents,
        theLocalVariables,
        theTreesTrapSignals);
    Transition aTransition = StateChartHelpers.createWA(aStatementState,
        priority++,
        theHandleState);
    aTransition.setLabel(
        this.theHandleEvents[0]
                .convertToKiel(anEsterelModule,
        theLocalEvents,
        theLocalVariables,
        null));
    StateChartHelpers.setImmediate(aTransition);
}
//No handler used
for (int i = 0; i < this.pureHandle.length; i++) {
    if (!this.pureHandle[i]) {
        WeakAbortion aTransition = StateChartHelpers
        .createWA(aStatementState,
            priority++,
            theNormalTermState);
        StateChartHelpers.setTrigger(aTransition,
            theEvents[i]);
        StateChartHelpers.setImmediate(aTransition);
        System.out.println(aTransition.getLabel().toString());
    }
}
 //niedrigste Prioritaet
StateChartHelpers.createNT(aStatementState,
    priority++,
    theNormalTermState);
```

```
    if (theHandleState != null) {
        theHandleState.setParent(aState);
        aState.addSubnode(theHandleState);
    }
    theNormalTermState.setParent(aState);
    aState.addSubnode(theNormalTermState);
    anInitialArc.setSource(anInitialState);
    anInitialArc.setTarget(aStatementState);
    anInitialState.setParent(aState);
    aStatementState.setParent(aState);
    aState.addSubnode(aStatementState);
    aState.addSubnode(anInitialState);
    //we added all the localsignals to theLocalEvenets
    // now we delete them again
    while (theLocalEvents.size() != oldsize) {
        theLocalEvents.remove(theLocalEvents.size() - 1);
    }
    return aState;
} //public final kiel.dataStructure.State convertToKiel(

/**
 * Implemtents the abstrac methode from <code>EsterelStatement</code>
 * Parses an esterel statment in an <code>org.jdom.Document</code>
 * representation of an .exp file if you have not called the
 * constructor <code> EsterelStatement(EsterelParser theparser)</code>
 * use the methode <code>preParseEsterelStatement</code> before
 * <code>parseEsterelStatement</code>.<br>
 * Sideeffects: the class variables are set Changes
 * <code>EsterelParser</code> class variables
 *
 * @param theparser
 *                an instance of <code>EsterelParser</code>
 * @throws EsterelParserException
 *                is only delivered
 * @see kiel.fileInterface.esterel.EsterelParserException
 * @see EsterelStatement#EsterelStatement(EsterelParser theparser)
 * @see EsterelStatement#parseEsterelStatement(EsterelParser)
 * @see kiel.fileInterface.esterel.EsterelParser
 * @see org.jdom.Document
 */
public final void parseEsterelStatement(
       final EsterelParser theparser)
       throws EsterelParserException {
    //get all children
    List anElementList = DOMHelpers.getElements(theparser.getXMLDocument(),
        this.getXPathSearchString()
              + "[not(local-name(.)='EOV')]");
    int anElementIndex = 0;
    if (anElementList.size() > 0
          && ((Element) anElementList.get(anElementIndex)).getName()
                .compareTo("NULL") == 0) {
        throw new EsterelParserException(
              this.getEsterelStatementName()
                    + " no Statement found");
    }
    theparser.incStatementCounter(); // we have
    // a
    // statement
    anElementIndex++;

if (anElementList.size() > 1
       && ((Element) anElementList.get(anElementIndex)).getName()
             .compareTo("NULL") == 0) {
    throw new EsterelParserException(
          this.getEsterelStatementName()
                + " no exception declaration found ");
}
    List anExceptionDeclarationReferrenceList =
        DOMHelpers.getElements(theparser.getXMLDocument(),
        "//*[@id='"
              + ((Element) anElementList.get(anElementIndex))
              .getAttributeValue("id")
              + "']/*[not(local-name(.)='EOV')]"
              + "[not(local-name(.)='SymbolTable')]"
              + "[not(local-name(.)='NULL')]");
    this.myExceptionSignals
    = new EsterelSignal[anExceptionDeclarationReferrenceList.size()];
    int emptyoffset = 0;
    for (int i = 0;
    i < anExceptionDeclarationReferrenceList.size();
    i++) {
         this.myExceptionSignals[i - emptyoffset] =
           theparser.getEsterelModule(
               theparser.getactualEsterModule())
               .getEsterelSignalByID(
                   ((Element) anExceptionDeclarationReferrenceList
                         .get(i))
                   .getAttributeValue("id"));
        if (this.myExceptionSignals[i - emptyoffset] == null) {
            emptyoffset ++;
            EsterelSignal[] dummy =
                new EsterelSignal[this.myExceptionSignals.length - 1];
            for (int d = 0; d < i; d++) {
                dummy[d] = this.myExceptionSignals[d];
            }
            this.myExceptionSignals = dummy;
        }
        if (this.myExceptionSignals.length < 1) {
            throw new EsterelParserException(
                  this.getEsterelStatementName()
                        + ": Found no Exceptionsignal");
        }
    } //for
    this.pureHandle =
        new boolean[this.myExceptionSignals.length];
    for (int i = 0; i < this.myExceptionSignals.length; i++) {
        this.pureHandle[i] = false;
    }
    anElementIndex++;

if (anElementList.size() > 2
       && ((Element) anElementList.get(anElementIndex)).getName()
```

```
                .compareTo("NULL") != 0) {
        theparser.incStatementCounter(anElementList.size()
                - anElementIndex);
        this.myHandleStatements = new int[anElementList.size()
                - anElementIndex];
        this.theHandleEvents = new
        EsterelSignalExpression[anElementList.size()
                - anElementIndex];
    } //else
} //parseEsterelStatement;

/**
 * Fills the substatements in the esterel module. <br>
 * Sideeffects: changes EsterelModule
 *
 * @param theparser
 *                the actual parser
 * @throws EsterelParserException
 *                is only delivered
 * @see kiel.fileInterface.esterel.EsterelParserException
 * @see kiel.fileInterface.esterel.EsterelParser
 * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
 */
public final void setSubStatments(
        final EsterelParser theparser)
        throws EsterelParserException {
    //get all children
    List anElementList =
        DOMHelpers.getElements(theparser.getXMLDocument(),
        this.getXPathSearchString()
                + "[not(local-name(.)='EOV')]");
    int anElementIndex = 0;
    if (anElementList.size() > 0
            && ((Element) anElementList.get(anElementIndex)).getName()
                    .compareTo("NULL") == 0) {
        throw new EsterelParserException(
                this.getEsterelStatementName()
                        + " no Statement found");
    }
        this.theTrapedStatement = theparser.getAddAt();
        theparser.getEsterelModule(theparser.getactualEsterModule())
                .addStatementToModule(theparser.getAddAt(),
                    theparser.createStatement(((Element)
                            (anElementList.get(anElementIndex)))));
        anElementIndex++;

    anElementIndex++; // declarations are parsed
    if (this.myHandleStatements != null) {
        //anElementIndex indexes the first PredicatedStatement element
        List theHandlerList =
            DOMHelpers.getElements(theparser.getXMLDocument(),
            this.getXPathSearchString()
                    + "[local-name(.)='"
                    + ((Element) anElementList.get(anElementIndex))
                    .getName()
                    + "']");
for (int theHandleIndex = 0;
theHandleIndex < theHandlerList.size();
theHandleIndex++) {
    if (((Element) theHandlerList.get(theHandleIndex)).getName()
            .compareTo("PredicatedStatement") == 0) {
        String aPredicatedStatementXPathSearchString =
            this.getXPathSearchString()
                + "[local-name(.)='"
                + ((Element) theHandlerList.get(theHandleIndex))
                .getName()
                + "'][@id='"
                + ((Element) theHandlerList.get(theHandleIndex))
                .getAttributeValue("id")
                + "']/*";
        List aPredicatedElementList = DOMHelpers.getElements(
            theparser.getXMLDocument(),
            aPredicatedStatementXPathSearchString);
        if (aPredicatedElementList.size() == 2) {
            this.myHandleStatements[theHandleIndex] =
                theparser.getAddAt();
            theparser
            .getEsterelModule(theparser.getactualEsterModule())
                    .addStatementToModule(theparser.getAddAt(),
                        theparser.createStatement(((Element)
                                (aPredicatedElementList.get(0)))));
            this.theHandleEvents[theHandleIndex] =
                new EsterelSignalExpression(
                    theparser,
                    "//*[@id='"
                            + ((Element) (aPredicatedElementList
                                    .get(1)))
                            .getAttributeValue("id")
                            + "']/*", theparser.getXMLDocument());
            if (this.theHandleEvents[theHandleIndex]
                                    .isSignal()) {
                String aSignalName =
                    this.theHandleEvents[theHandleIndex]
                                .getMySignal()
                                .getSignalIdentifier();
                for (int i = 0;
                i < this.myExceptionSignals.length;
                i++) {
                    if (this.myExceptionSignals[i]
                            .getSignalIdentifier()
                            .compareTo(aSignalName) == 0) {
                        this.pureHandle[i] = true;
                    }
                }
            }
        } else {
            this.myHandleStatements[theHandleIndex] = -1;
            throw new EsterelParserException(
                    this.getEsterelStatementName()
                            + " exceptionhandler not found");
        }
    } //if
```

```
            } //for
        } //if
    } //setsubstatements

    /**
     * Implemtents the abstrac methode from <code>EsterelStatement</code>
     * and Returns a string representation of an esterel statement. <br>
     * Sideeffects: none
     *
     * @param anEsterelModule
     *                an esterel modul representation
     * @return a String representation of an esterel statment
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
     * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
     */
    public final String toString(
            final EsterelModule anEsterelModule) {
        String aExceptionList = "";
        for (int i = 0; i < this.myExceptionSignals.length; i++) {
            aExceptionList +=
                this.myExceptionSignals[i]
                 .toString(anEsterelModule);
            aExceptionList += "\n";
        }
        String result = this.getEsterelStatementName()
                + " "
                + aExceptionList
                + " in\n"
                + ((EsterelStatement) anEsterelModule.getEsterelProgram()
                        .get(this.theTrapedStatement)).toString(anEsterelModule)
                + "\n";
        if (this.myHandleStatements != null) {
            for (int i = 0; i < this.myHandleStatements.length; i++) {
                result += "handel "
                        + this.theHandleEvents[i].toString(anEsterelModule)
                        + "do\n"
                        + ((EsterelStatement) anEsterelModule
                                .getEsterelProgram()
                                .get(this.myHandleStatements[i]))
                                .toString(anEsterelModule)
                        + "\n";
            }
            // for
        } //if
        result += "end "
                + this.getEsterelStatementName() + "\n";
        return result;
    } //public final String toString(final EsterelModule anEsterelModule)
    // {
    /**
     * @return Returns the myExceptionDeclarationList.
     */
    public final EsterelSignal[] getMyExceptionDeclarationList() {
        return this.myExceptionSignals;
    }
    /**
     * @return Returns the myHaltSignal.
     */
    public final EsterelSignal getMyHaltSignal() {
        return this.myHaltSignal;
    }
}
```

C.3. Paket kiel.optimizer

Das Paket kiel.optimizer enthält die Klassen

- OptimizerChooser, welche den zu verwendenen Optimierer auswählt,
- Optimizer, welche einen Optimierungs-Algorithmus abstrakt definiert,
- OptimizationRule, welche die Optimierungsregeln abstrakt definiert,
- OptimizerProperties, welche Einstellungsmöglichkeiten bei der Optimierung vorgibt,
- OptimizerException, welche in Ausnahmefällen als *Exception* geworfen wird.

C.3.1. OptimizationRule

Auflistung C.50: Die Klasse `OptimizationRule`

```
package kiel.optimizer;

import kiel.dataStructure.CompositeState;
import kiel.dataStructure.StateChart;

/**
 * <p>
 *  An abstract class for optimization rules.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.25 $ last modified $Date: 2006/02/06 18:36:04 $
 */
public abstract class OptimizationRule {
    /**
     * the rule name.
     */
    private String ruleName = "";

    /**
     * a statechart.
     */
    private StateChart theChart = null;

    /**
     * @param aStateChart
     *                  a statechart.
     */
    public OptimizationRule(final StateChart aStateChart) {
        super();
        this.theChart = aStateChart;
    }

    /**
     * @return Returns the ruleName.
     */
    public final String getRuleName() {
        return this.ruleName;
    }


    /**
     * checks if a optimization for <code>aState</code> is possible. Has
     * to be implemented in inherited classes.
     *
     * @param aState
     *                  a composite state
     * @return true if optimization is possible else false.
     * @see kiel.dataStructure.CompositeState
     */
    public abstract boolean isOptimizationPossible(final CompositeState aState);

    /**
     * optimization for <code>aState</code>. Has to be implemented in
     * inherited classes.
     *
     * @param aState
     *                  a composite state
     * @param aStateChart the statechart.
     * @return a boolean true if something was optimized else false.
     * @throws OptimizerException
     *                  not throw, may be trown in called methods.
     * @see kiel.dataStructure.CompositeState
     */
    public abstract boolean optimizeState(final CompositeState aState,
           final StateChart aStateChart)
           throws OptimizerException;

    /**
     * @return the rule name.
     */
    public final String toString() {
        return this.ruleName;
    }

    /**
     * sets the rule name.
     *
     * @param aName
     *                  a name.
     */
    protected final void setRuleName(final String aName) {
        this.ruleName = aName;
    }
    /**
     * @return Returns the theChart.
     */
    public final StateChart getStateChart() {
        return this.theChart;
    }
    /**
     * @param aChart The theChart to set.
     */
    public final void setStateChart(final StateChart aChart) {
        this.theChart = aChart;
    }
}
```

C.3.2. OptimizerChooser

Auflistung C.51: Die Klasse `OptimizerChooser`

```
package kiel.optimizer;

import kiel.dataStructure.StateChart;
import kiel.optimizer.esterelstudio.EsterelStudioChartOptimizer;

/**
 * <p>
 *  This static class chooses a optimizer
 * for a statechart. At the moment only a EsterelStudioChart
 * optimizer is availble..
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.22 $ last modified $Date: 2006/02/07 16:59:47 $
 */
public final class OptimizerChooser {

    /**
     * standart constructor.
     */
    private OptimizerChooser() {
        super();
    }
    /**
     * abstract methode. starts the optimization of an statechart.
     *
     * @param all
     *                if true then all levels will be optimized else only the
     *                lowest unoptimized level .
     * @param aChart
     *                a statechart
     * @return a statechart
     * @throws OptimizerException only delivers the exception.
     */
    public static StateChart startOptimize(
            final StateChart aChart,
            final boolean all) throws OptimizerException {
        OptimizerProperties.load();
        if (aChart != null) {
            Optimizer theOptimizer = null;
            if (aChart.getModelSource().compareTo("Esterel Studio") == 0) {
                theOptimizer = new EsterelStudioChartOptimizer();
                return theOptimizer.startOptimize(aChart, all);
                }
        }
        return aChart;
    }
}
```

C.3.3. OptimizerException

Auflistung C.52: Die Klasse `OptimizerException`

```
package kiel.optimizer;

/**
 * <p>
 * A class for Optimizer exceptions.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.25 $ last modified $Date: 2006/02/06 18:36:04 $
 */
public class OptimizerException extends Exception {
    /**
     * standart constructor.
     */
    public OptimizerException() {
        super();
    }

    /**
     * @param arg0
     *                a string
     */
    public OptimizerException(final String arg0) {
        super(arg0);
    }

    /**
     * @param arg0
     *                a throwable
     */
    public OptimizerException(final Throwable arg0) {
        super(arg0);
    }

    /**
     * @param arg0
     *                a string
     * @param arg1
     *                athrowabel
     */
    public OptimizerException(final String arg0, final Throwable arg1) {
        super(arg0, arg1);
    }
}
```

C.3.4. Optimizer

Auflistung C.53: Die Klasse `Optimizer`

```
package kiel.optimizer;

import kiel.dataStructure.StateChart;

/**
 * <p>
 *  An abstract class for statechart optimizer.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.31 $ last modified $Date: 2006/02/06 18:36:04 $
 */
public abstract class Optimizer {
    /**
     * the statechart.
     */
    private StateChart aStateChart = null;

    /**
     * Standart constructor.
     */
    public Optimizer() {
        this(
                null);
    }

    /**
     * constructor which sets a statechart.
     *
     * @param aChart
     *          a KIEL StateChart.
     */
    public Optimizer(final StateChart aChart) {
        super();
        this.setStateChart(aChart);
    }

    /**
     * @return Returns the aStateChart.
     */
    public final StateChart getStateChart() {
        return this.aStateChart;
    }

    /**
     * @param stateChart
     *          The aStateChart to set.
     */
    public final void setStateChart(
            final StateChart stateChart) {
        this.aStateChart = stateChart;
    }

    /**
     * abstract methode. starts the optimization of an statechart.
     *
     * @param all
     *          if true then all levels will be optimized else only the
     *          lowest unoptimized level .
     * @param aChart
     *          a statechart
     * @throws OptimizerException
     *           if something went wrong.
     * @return a statechart
     */
    public abstract StateChart startOptimize(
            final StateChart aChart,
            final boolean all)
            throws OptimizerException;
}
```

C.3.5. OptimizerProperties

Auflistung C.54: Die Klasse `OptimizerProperties`

```
package kiel.optimizer;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;

/**
* <p>
*  Used to load and store user defineable properties.
* </p>
* <p>
* Copyright: Copyright (c) 2005
* </p>
* <p>
* Company: Uni Kiel
* </p>
*
* @author <a href="mailto:rtkiel@informatik.uni-kiel.de">rtkiel </a>
* @version $Revision: 1.6 $
*/
public final class OptimizerProperties {
       /**
     * The property key for debug mode.
     */
    private static final String DEBUGMODE = "DebugMode";

    /**
     * The default ressource for the properties.
     */
    private static final String DEFAULTRESOURCE = "optimizer.properties";
    /**
     * The conditional optimization property key .
     */
    private static final String
    OPTIMIZEESTELIMINATENEEDLESSCONDITIONAL =
        "OptimizeESTEliminateNeedlessConditional";
    /**
     * The or state optimization property key .
     */
    private static final String
    OPTIMIZEESTELIMINATENEEDLESSNORMALTRANSITIONSL =
        "OptimizeESTEliminateNeedlessNormalTransitions";

    /**
     * The or state optimization property key .
     */
    private static final String
    OPTIMIZEESTELIMINATENEEDLESSSIMPLESTATES =
        "OptimizeESTEliminateNeedlessSimpleStates";
    /**
     * The or state optimization property key .
     */
    private static final String
    OPTIMIZEESTELIMINATENOTABORTEDNWITHOUTLOCALORSTATES =
        "OptimizeESTEliminateNotAbortednWithoutLocalsOrStates";
    /**
     * The or state optimization property key .
     */
    private static final String
    OPTIMIZEESTJOINFINALSTATES =
        "OptimizeESTJoinFinalStates";

    /**
     * The or state optimization property key .
     */
    private static final String OPTIMIZEESTORSTATESW2SUBS =
        "OptimizeESTOrStatesWithTwoSubstates";

    /**
     * The final state optimization property key .
     */
    private static final String OPTIMIZEESTUPDATEFINALSTATES =
        "OptimizeESTUpdateFinalStates";

    /**
     * These are the internal properties.
     */
    private static Properties properties;

    /**
     * This is the key for user specific file.
     */
    private static final String PROPERTIESFILE = System.getProperty("user.home")
           + File.separator
           + ".kiel"
           + File.separator
           + DEFAULTRESOURCE;
       /**
     * @return true, if debugmode is true.
     */
    public static boolean getDebugMode() {
        return properties.getProperty(DEBUGMODE)
                .equalsIgnoreCase("true");
    }
    /**
     * @return true, if option is set to true.
     */
    public static boolean getOptimizeESTEliminateNeedLessConditional() {
        return properties.getProperty(
            OPTIMIZEESTELIMINATENEEDLESSCONDITIONAL)
```

```
            .equalsIgnoreCase("true");
    }
    /**
     * @return true, if option is set to true.
     */
    public static boolean getOptimizeESTEliminateNeedlessNormalTransitions() {
        return properties.getProperty(
            OPTIMIZEESTELIMINATENEEDLESSNORMALTRANSITIONSL)
                .equalsIgnoreCase("true");
    }

    /**
     * @return true, if option is set to true.
     */
    public static boolean getOptimizeESTEliminateNeedlessSimpleStates() {
        return properties.getProperty(
            OPTIMIZEESTELIMINATENEEDLESSSIMPLESTATES)
                .equalsIgnoreCase("true");
    }
    /**
     * @return true, if option is set to true.
     */
    public static boolean getOptimizeESTJoinFinalStates() {
        return properties.getProperty(
            OPTIMIZEESTJOINFINALSTATES)
                .equalsIgnoreCase("true");
    }
    /**
     * @return true, if option is set to true.
     */
    public static boolean getOptimizeESTOrStatesW2Subs() {
        return properties.getProperty(OPTIMIZEESTORSTATESW2SUBS)
                .equalsIgnoreCase("true");
    }
    /**
     * @return true, if option is set to true.
     */
    public static boolean getOptimizeESTUpdateFinalStates() {
        return properties.getProperty(
            OPTIMIZEESTUPDATEFINALSTATES)
                .equalsIgnoreCase("true");
    }

     /**
     * @return true, if option is set to true.
     */
    public static boolean
    optimizeESTEliminateNotAbortednWithoutLocalsOrStates() {
        return properties.getProperty(
            OPTIMIZEESTELIMINATENOTABORTEDNWITHOUTLOCALORSTATES)
                .equalsIgnoreCase("true");
    }

    /**
     * Copies properties to user specific file.
     *
     * @return true, if copy was successful
     */
    private static boolean copyDefaults() {
        InputStream is;
        FileWriter fw;
        boolean success = true;
        try {
            is = OptimizerProperties.class.getResourceAsStream(DEFAULTRESOURCE);
            fw = new FileWriter(
                    PROPERTIESFILE);
            int c = is.read();
            while (c >= 0) {
                fw.write(c);
                c = is.read();
            }
            fw.flush();
            is.close();
            fw.close();
        } catch (IOException storeExeception) {
            success = false;
        }
        return success;
    }

    /**
     * Loads the defaults form resource file.
     *
     * @return the default values
     */
    private static Properties getDefaults() {
        Properties defaults = new Properties();
        try {
            defaults.load(OptimizerProperties.class.
                getResourceAsStream(DEFAULTRESOURCE));
        } catch (IOException e) {
            defaults.clear();
        }
        return defaults;
    }

    /**
     * Loads properties from user specific file.
     *
     * @return true, if loading was successful
     */
    private static boolean loadProperties() {
        boolean success = true;
        try {
            properties.load(new FileInputStream(
                    PROPERTIESFILE));
        } catch (IOException loadException) {
            success = false;
        }
        return success;
    }
```

```
    /**
     * This method will load the properties from local file. If this file
     * does not exist, the defaults are written to local file.
     *
     * @return true, if loading was successful
     */
    protected static boolean load() {
        boolean success;
        properties = new Properties(
                getDefaults());
        if (!loadProperties()) {
            success = copyDefaults();
        } else {
            success = true;
        }
        return success;
    }

    /**
     * Reloads the properties file.
     */
    protected static void reload() {
        loadProperties();
    }

    /**
     * This is a util class.
     */
    private OptimizerProperties() {
        super();

    }
}
```

C.4. Paket kiel.optimizer.esterelstudio

Das Paket kiel.optimizer.esterelstudio implementiert den Optimierungs-Algorithmus aus Abschnitt 6.4 in der Klasse EsterelStudioChartOptimizer, welche in der Abbildung C.4 dargestellt ist. Zumdem werden hier die Optimierungsregeln aus Kapitel 4 implementiert. Der Aufbau der Klassen der Optimierungsregeln ist in der Abbildung C.4 dargestellt.

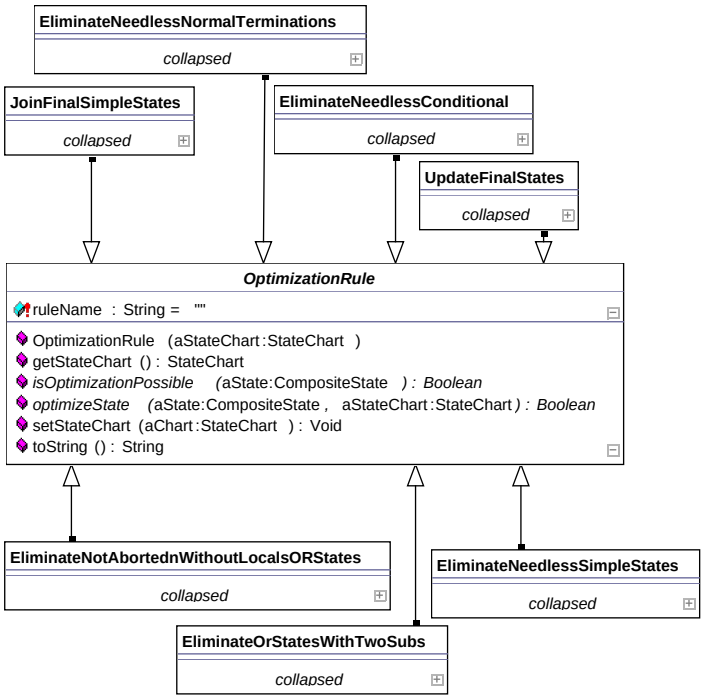


Abbildung C.37.: Klassediagramm OptimizerRules

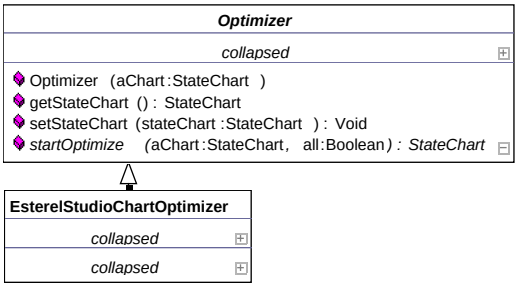


Abbildung C.38.: Klassediagramm EsterelStudioChartOptimizer

C.4.1. EliminateNeedlessConditional

Auflistung C.55: Die Klasse `EliminateNeedlessConditional`

```
package kiel.optimizer.esterelstudio;

import java.util.ArrayList;
import kiel.dataStructure.ANDState;
import kiel.dataStructure.CompositeState;
import kiel.dataStructure.CompoundLabel;
import kiel.dataStructure.DynamicChoice;
import kiel.dataStructure.StateChart;
import kiel.dataStructure.Transition;
import kiel.dataStructure.TransitionLabel;
import kiel.dataStructure.action.Action;
import kiel.dataStructure.action.Actions;
import kiel.optimizer.OptimizationRule;
import kiel.optimizer.OptimizerException;
import kiel.util.CompoundLabelParser;
import kiel.util.StateChartHelpers;

/**
 * <p>
 *  Eliminates conditionals with only the default transition.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.9 $ last modified $Date: 2006/02/17 20:00:22 $
 */
public class EliminateNeedlessConditional
       extends OptimizationRule {
   /**
    * @param aChart
    *                aStateChart.
    */
   public EliminateNeedlessConditional(
          final StateChart aChart) {
      super(
            aChart);
      super.setRuleName("EliminateNeedlessConditional");
   }

   /**
    * checks if a optimization for <code>aState</code> is possible.
    *
    * @param aState
    *                a composite state
    * @return true if optimization is possible else false.
    * @see kiel.dataStructure.CompositeState
    */
   public final boolean isOptimizationPossible(
          final CompositeState aState) {
      if (aState instanceof ANDState) {
         return false;
      }
      return true;
   }

   /**
    * optimization for <code>aState</code>.
    *
    * @param aState
    *                a composite state
    * @param aStateChart a statechart
    * @return null
    * @throws OptimizerException
    *                 not throw, may be trown in called methods.
    * @see kiel.dataStructure.CompositeState
    */
   public final boolean optimizeState(
          final CompositeState aState,
          final StateChart aStateChart)
          throws OptimizerException {
      boolean result = false;
      //this.setStateChart(aStateChart);
      if (isOptimizationPossible(aState)) {
         // test for useless dynamic choices
         ArrayList pseudoStates = StateChartHelpers
                .getSubPseudoStates(aState);
         for (int i = 0; i < pseudoStates.size(); i++) {
            if (pseudoStates.get(i) instanceof DynamicChoice) {
               DynamicChoice dc = (DynamicChoice) pseudoStates.get(i);
               if (dc.getOutgoingTransitions().size() == 1
                      && (StateChartHelpers.getEffect(
                             ((Transition) ((ArrayList) dc
                                    .getOutgoingTransitions())
                                    .get(0)).getLabel(),
                             aStateChart,
                             aState).getActions().length == 0 || (dc
                             .getIncomingTransitions()
                                 .size() == 1))) {
                  result = true;
                  boolean abort = false;
                  TransitionLabel thedcLabel =
                      ((Transition) ((ArrayList) dc
                          .getOutgoingTransitions()).get(0))
                          .getLabel();
                  Action[] choiceAction = null;
                  if (thedcLabel instanceof CompoundLabel) {
                      choiceAction = ((CompoundLabel) thedcLabel)
```

```
            .getEffect()
                .getActions();
    } else {
        String aString = thedcLabel.toString();
        String actionString = "";
        if (aString.lastIndexOf("/") > -1) {
           actionString = aString
                .substring(
                    aString.lastIndexOf("/") + 1,
                    aString.length());
        }
        try {
            CompoundLabelParser.getInstance();
            CompoundLabelParser.setStateChart(aStateChart);
            CompoundLabelParser.setCompositeState(aState);
            choiceAction = CompoundLabelParser
            .parseActions(actionString)
            .getActions();
        } catch (Exception e) {
            choiceAction = new Action[0];
            if (actionString.length() > 0) {
                abort = true;
            }
        }
        CompoundLabelParser.setCompositeState(null);
    }
    if (!abort) {
    ArrayList allIncomming = (ArrayList) dc
            .getIncomingTransitions();
    for (int j = 0; j < allIncomming.size(); j++) {
        Transition aITrans = (Transition) allIncomming
                .get(j);
        TransitionLabel aLabel = aITrans.getLabel();
        Action[] inActions = null;
        if (aLabel instanceof CompoundLabel) {
            inActions = ((CompoundLabel) aLabel)
                    .getEffect()
                        .getActions();
        } else {
            String aString = aLabel.toString();
            String actionString = "";
            if (aString.lastIndexOf("/") > 0) {
               actionString = aString
                    .substring(
                        aString.lastIndexOf("/") + 1,
                        aString.length());
            }
            try {
                CompoundLabelParser.getInstance();
                CompoundLabelParser.setStateChart(aStateChart);
                CompoundLabelParser.setCompositeState(aState);
                inActions =
                    CompoundLabelParser.parseActions(
                    actionString)
                        .getActions();
            } catch (Exception e) {
                inActions = new Action[0];
                if (actionString.length() > 0) {
                    abort = true;
                }
            }
        }
        if (!abort) {
        // join Actions Actions
        Action[] joinedAction = new Action[inActions.length
                + choiceAction.length];
        for (int k = 0; k < inActions.length; k++) {
            joinedAction[k] = inActions[k];
        }
        for (int k = 0; k < choiceAction.length; k++) {
            joinedAction[k
                    + inActions.length] = choiceAction[k];
        }
        if (aLabel instanceof CompoundLabel) {
            CompoundLabel aCl = ((CompoundLabel) aLabel);
            aCl.setEffect(new Actions(
                    joinedAction));
            aITrans.setLabel(aCl);
        } else if (aLabel.toString().compareTo(
                "") == 0) {
            CompoundLabel aCl = new CompoundLabel();
            aCl.setEffect(new Actions(
                    joinedAction));
            aCl.setIDManual(aITrans.getLabel().getID());
            aITrans.setLabel(aCl);
        }
        aITrans.setTarget(((Transition) ((ArrayList) dc
                .getOutgoingTransitions()).get(0))
                .getTarget());
        }
    } //for j
    while (dc.getIncomingTransitions().size() > 0) {
        dc.removeIncomingTransition(
                (Transition) ((ArrayList) dc
                        .getIncomingTransitions())
                        .get(0));
    }
    while (dc.getOutgoingTransitions().size() > 0) {
       Transition aT = ((Transition)
               ((ArrayList) dc.getOutgoingTransitions())
               .get(0));
       aT.getTarget().removeIncomingTransition(aT);
       dc.removeOutgoingTransition(aT);
    }
    dc.getParent().removeSubnode(dc);
    }
  } //if
}

} //for i
}
return result;
```

}

}

C.4.2. EliminateNeedlessNormalTerminations

Auflistung C.56: Die Klasse `EliminateNeedlessNormalTerminations`

```
package kiel.optimizer.esterelstudio;

import java.util.ArrayList;

import kiel.dataStructure.*;
import kiel.optimizer.OptimizationRule;
import kiel.optimizer.OptimizerException;
import kiel.util.StateChartHelpers;

/**
 * <p>
 *  Eliminates needless normal terminations.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.5 $ last modified $Date: 2006/02/17 17:47:19 $
 */
public class EliminateNeedlessNormalTerminations
        extends OptimizationRule {
    /**
     * @param aChart
     *                 aStateChart.
     */
    public EliminateNeedlessNormalTerminations(final StateChart aChart) {
        super(
                aChart);
        super.setRuleName("EliminateNeedlessNormalTerminations");
    }

    /**
     * checks if a optimization for <code>aState</code> is possible.
     *
     * @param aState
     *                 a composite state
     * @return true if optimization is possible else false.
     * @see kiel.dataStructure.CompositeState
     */
    public final boolean isOptimizationPossible(
            final CompositeState aState) {
        if (aState instanceof ORState && aState.getParent() != null) {
            if (StateChartHelpers.getSubFinalStates(aState).size() == 0) {
            return true;
            }
        } else if (aState instanceof ANDState && aState.getParent() != null) {
            ArrayList allRegions = ((ANDState) aState).getSubnodes();
            for (int i = 0; i < allRegions.size(); i++) {
                if (StateChartHelpers
                        .getSubFinalStates(
                                (CompositeState) allRegions
                                .get(i))
                                .size() == 0) {
                    return true;
                }
            }
        }
        return false;
    }

    /**
     * optimization for <code>aState</code>.
     *
     * @param aState
     *                 a composite state
     * @param aStateChart
     *                 a statechart
     * @return true is something has changed.
     * @throws OptimizerException
     *                 not throw, may be trown in called methods.
     * @see kiel.dataStructure.CompositeState
     */
    public final boolean optimizeState(
            final CompositeState aState,
            final StateChart aStateChart)
            throws OptimizerException {
        boolean result = false;
        this.setStateChart(aStateChart);
        if (isOptimizationPossible(aState)) {
            // if there is more than one final simple state.
            if (StateChartHelpers
                    .getOutgoingNormalTerminations(aState)
                    .size() > 0) {
                NormalTermination aNT = (NormalTermination)
                StateChartHelpers
                .getOutgoingNormalTerminations(aState)
                        .get(0);
                aNT.getTarget().removeIncomingTransition(aNT);
                aNT.getSource().removeOutgoingTransition(aNT);
                StateChartHelpers.eliminateDeadStates(
                    aState.getParent());
                result = true;
            }
        }
        return result;
    }
}
```

C.4.3. EliminateNeedlessSimpleStates

Auflistung C.57: Die Klasse `EliminateNeedlessSimpleStates`

```
package kiel.optimizer.esterelstudio;

import java.util.ArrayList;

import kiel.dataStructure.CompositeState;
import kiel.dataStructure.CompoundLabel;
import kiel.dataStructure.Node;
import kiel.dataStructure.SimpleState;
import kiel.dataStructure.State;
import kiel.dataStructure.StateChart;
import kiel.dataStructure.Transition;
import kiel.optimizer.OptimizationRule;
import kiel.optimizer.OptimizerException;
import kiel.util.StateChartHelpers;

/**
 * <p>
 *  Eliminates needless simple states.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.3 $ last modified $Date: 2006/02/06 18:36:04 $
 */
public class EliminateNeedlessSimpleStates
       extends OptimizationRule {
   /**
    * @param aChart
    *                astatechart
    */
   public EliminateNeedlessSimpleStates(
          final StateChart aChart) {
      super(
             aChart);
      super.setRuleName("EliminateNeedlessSimpleStates");
   }

   /**
    * checks if a optimization for <code>aState</code> is possible.
    *
    * @param aState
    *                a composite state
    * @return true if optimization is possible else false.
    * @see kiel.dataStructure.CompositeState
    */
   public final boolean isOptimizationPossible(
          final CompositeState aState) {
      return false;
   }

   /**
    * checks if a optimization for <code>aState</code> is possible.
    *
    * @param aState
    *                a node
    * @return true if optimization is possible else false.
    * @see kiel.dataStructure.CompositeState
    */
   public final boolean isOptimizationPossible(
          final Node aState) {
      if (aState instanceof SimpleState
            && ((SimpleState) aState).getDoActivity()
            .getActions().length == 0
            && ((SimpleState) aState).getEntry().getActions().length == 0
            && ((SimpleState) aState).getExit().getActions().length == 0
            && aState.getIncomingTransitions().size() > 0
            && aState.getOutgoingTransitions().size() > 0
            && (aState.getIncomingTransitions().size() == 1 || aState
                  .getOutgoingTransitions()
                     .size() == 1)) {
               return true;
            }
      return false;
   }

   /**
    * optimization for <code>aState</code>.
    *
    * @param aState
    *                a composite state
    * @param aStateChart
    *                a StateChart
    * @return true if something happed, else false
    * @throws OptimizerException
    *                not throw, may be trown in called methods.
    * @see kiel.dataStructure.CompositeState
    */
   public final boolean optimizeState(
          final CompositeState aState,
          final StateChart aStateChart)
          throws OptimizerException {
      boolean result = false;
      boolean startAgain = true;
      this.setStateChart(aStateChart);
      while (startAgain) {
         startAgain = false;
         ArrayList subStates = StateChartHelpers.getSubStates(aState);
```

```
boolean[] fusion = new boolean[subStates.size()];
for (int i = 0; i < subStates.size(); i++) {
    fusion[i] = true;
    if (this.isOptimizationPossible((Node) subStates.get(i))) {
        if (((Node) subStates.get(i))
                .getIncomingTransitions()
                .size() == 1) {
            Transition useless = (Transition) ((ArrayList)
                    ((Node) subStates
                    .get(i)).getIncomingTransitions()).get(0);
            ArrayList allOutTrans = (ArrayList) ((Node) subStates
                    .get(i)).getOutgoingTransitions();
            for (int j = 0; j < allOutTrans.size(); j++) {
                fusion[i] = fusion[i]
                        && this.isFusionable(
                                useless,
                                (Transition) allOutTrans.get(j),
                                aState);
            }
            if (fusion[i]) {
                StateChartHelpers.ensurePriority(
                        useless.getSource(),
                        allOutTrans.size(),
                        useless.getPriority().getValue());
                boolean abort = false;
                while (allOutTrans.size() > 0 && !abort) {
                    Transition newTrans = (StateChartHelpers
                            .combineTransitions(
                                    useless,
                                    (Transition) allOutTrans
                                            .get(0),
                                    false,
                                    this.getStateChart(),
                                    aState));
                    if (newTrans != null) {
                        newTrans.setPriority(useless
                                .getPriority()
                                    .getValue()
                                + ((Transition) allOutTrans
                                        .get(0))
                                        .getPriority()
                                            .getValue() - 1);
                        newTrans
                                .getPriority()
                                    .setIDManual(
                                     ((Transition) allOutTrans
                                                  .get(0))
                                                  .getPriority()
                                                     .getID());
                        ((Transition) allOutTrans.get(0))
                                .getTarget()
                                    .removeIncomingTransition(
                                      ((Transition) allOutTrans
                                                  .get(0)));
                        ((Transition) allOutTrans.get(0))
                                .getSource()
                                        .removeOutgoingTransition(
                                  ((Transition) allOutTrans
                                                  .get(0)));
                    } else {
                        abort = true;
                    }
                } //while
                if (!abort) {
                useless.getSource().removeOutgoingTransition(
                        useless);
                useless.getTarget().removeIncomingTransition(
                        useless);
                startAgain = true;
                result = true;
               }
            }
        } else {
            Transition useless = (Transition) ((ArrayList)
                    ((Node) subStates
                    .get(i)).getOutgoingTransitions()).get(0);
            ArrayList allInTrans = (ArrayList) ((Node) subStates
                    .get(i)).getIncomingTransitions();
            for (int j = 0; j < allInTrans.size(); j++) {
                fusion[i] = fusion[i]
                        && this
                                .isFusionable(
                                        (Transition) allInTrans
                                                .get(j),
                                    (Transition) ((ArrayList)
                                            ((Node) subStates
                                                .get(i))
                                        .getOutgoingTransitions())
                                                .get(0),
                                        aState);
            }
            if (fusion[i]) {
                boolean abort = false;
                while (allInTrans.size() > 0 && !abort) {
                    Transition newTrans = (StateChartHelpers
                            .combineTransitions(
                                    (Transition) allInTrans
                                            .get(0),
                                    useless,
                                    true,
                                    this.getStateChart(),
                                    aState));
                    if (newTrans != null) {
                        newTrans
                                .setPriority(
                                        ((Transition) allInTrans
                                        .get(0)).getPriority());
                        ((Transition) allInTrans.get(0))
                                .getSource()
                                    .removeOutgoingTransition(
                                            ((Transition) allInTrans
                                                    .get(0)));
```

```
        ((Transition) allInTrans.get(0))
                .getTarget()
                .removeIncomingTransition(
                        ((Transition) allInTrans
                                .get(0)));
220
            } else {
                abort = true;
                }
            }
            if (!abort) {
            useless.getTarget().removeIncomingTransition(
                useless);
            useless.getSource().removeOutgoingTransition(
                useless);
            startAgain = true;
230         result = true;
            }
          }
        }
      } // substates
    }
    if (result) {
      StateChartHelpers.eliminateDeadStates(aState);
    }
    return result;
240 }

    /**
     * checks if the two transtion are fusionable.
     *
     * @param a         incommingTransition
     * @param b         outgoing Transition
     *
250  * @param aState    a composite state
     *
     * @return true if fusionabel
     */
    private boolean isFusionable(
            final Transition a,
            final Transition b,
            final CompositeState aState) {
        boolean result = false;
        if (a.getID().compareTo(
                b.getID()) != 0) {
            // trigger is the same and the first trans has no action
260         if ((StateChartHelpers.compareTriggerwithoutCounter(
                    StateChartHelpers.getTrigger(
                            a.getLabel(),
                            this.getStateChart(),
                            aState),
                    StateChartHelpers.getTrigger(
                            b.getLabel(),
                            this.getStateChart(),
                            aState))
                    && StateChartHelpers.getEffect(
                            a.getLabel(),
                            this.getStateChart(),
                            aState).getActions().length == 0
                    && !(StateChartHelpers.isDefaultTransition(a))
270                 && ((a.getSource() instanceof State
                        && ((State) a.getSource()).getEntry()
                            .getActions().length == 0
                        && ((State) a.getSource()).getExit()
                            .getActions().length == 0
                        && ((State) a.getSource()).getDoActivity()
                            .getActions().length == 0
                            )
                            || !(a.getSource() instanceof State))
                    )
                    || (b.getLabel() instanceof CompoundLabel
280                     && StateChartHelpers
                            .withoutTrigger((CompoundLabel) b.getLabel()))) {
                result = true;
            }
        }
        return result;
    }
}
```

C.4.4. EliminateNotAbortednWithoutLocalsORStates

Auflistung C.58: Die Klasse `EliminateNotAbortednWithoutLocalsORStates`

```
package kiel.optimizer.esterelstudio;

import java.util.ArrayList;
import kiel.dataStructure.ANDState;
import kiel.dataStructure.CompositeState;
import kiel.dataStructure.Node;
import kiel.dataStructure.ORState;
import kiel.dataStructure.StateChart;
import kiel.dataStructure.Transition;
import kiel.optimizer.OptimizationRule;
import kiel.optimizer.OptimizerException;
import kiel.util.StateChartHelpers;

/**
 * <p>
 *  Eliminates an ORState if it has no outgoing abotions and no
 * local declarations.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.4 $ last modified $Date: 2006/02/07 19:20:42 $
 */
public class EliminateNotAbortednWithoutLocalsORStates
        extends OptimizationRule {
    /*
     * number of min substates.
     */
    //   private int minNumberofSubStates = 40;
    /**
     * @param aChart
     *                aStateChart.
     */
    public EliminateNotAbortednWithoutLocalsORStates(final StateChart aChart) {
        super(
                aChart);
        super.setRuleName("EliminateNotAbortednWithoutLocalsORStates");
    }

    /**
     * checks if a optimization for <code>aState</code> is possible.
     *
     * @param aState
     *                a composite state
     * @return true if optimization is possible else false.
     * @see kiel.dataStructure.CompositeState
     */
    public final boolean isOptimizationPossible(
            final CompositeState aState) {
        if (aState.getParent() != null
            && StateChartHelpers.hasDeepHistory(aState)
            == StateChartHelpers
            .hasDeepHistory(aState.getParent())) {
         if (aState.getDoActivity().getActions().length == 0
                 && aState.getEntry().getActions().length == 0
                 && aState.getExit().getActions().length == 0
                 && aState.getLocalEvents().size() == 0
                 && aState.getVariables().size() == 0
                 && StateChartHelpers
                 .getIncommingSuspension(aState).size() == 0
                 && StateChartHelpers
                 .getOutgoingAbortions(aState).size() == 0) {
             if (/*
                 * (StateChartHelpers.getSubStates( aState).size() <
                 * this.minNumberofSubStates) &&
                 */aState instanceof ORState) {
                 return true;
             }
             if ((aState instanceof ANDState
                     && aState.getParent().getSubnodes().size()
                     == 2 && aState.getOutgoingTransitions()
                     .size() == 0)
                     && aState.getIncomingTransitions().size() > 0
                     && ((Transition) ((ArrayList) aState
                             .getIncomingTransitions())
                             .get(0))
                             .getLabel()
                             .toString()
                             .compareTo("") == 0) {
                 return false;
             }
         }
        }
        return false;
    }

    /**
     * optimization for <code>aState</code>.
     *
     * @param aState
     *                a composite state
     * @param aStateChart
     *                a <code>StateChart</code>
     * @return null
     * @throws OptimizerException
     *                not throw, may be trown in called methods.
     * @see kiel.dataStructure.CompositeState
```

```
 */
public final boolean optimizeState(
        final CompositeState aState,
        final StateChart aStateChart)
    throws OptimizerException {
  boolean result = false;
  this.setStateChart(aStateChart);
  if (isOptimizationPossible(aState)) {
    if (aState instanceof ORState) {
      result = true;
      StateChartHelpers
        .deleteOrStateButSaveSubStates((ORState) aState);
    } else if (aState instanceof ANDState) {
      if (aState.getParent().getParent() != null) {
        ArrayList in = (ArrayList) aState.getParent()
            .getIncomingTransitions();
        ArrayList out = (ArrayList) aState.getParent()
            .getOutgoingTransitions();
        while (in.size() > 0) {
          ((Transition) in.get(0)).setTarget(aState);
          aState.getParent()
              .removeIncomingTransition(
                  ((Transition) in.get(0)));
        }
        while (out.size() > 0) {
          ((Transition) out.get(0)).setSource(aState);
          aState.getParent()
              .removeOutgoingTransition(
                  ((Transition) out.get(0)));
        }
        CompositeState newParent = aState.getParent()
            .getParent();
        Node oldParent = aState.getParent();
        newParent.addSubnode(aState);
        newParent.removeSubnode(oldParent);
        aState.setParent(newParent);
        oldParent.setParent(null);
      } /*
        * else { State a = aState; String id
        * =aState.getParent().getID(); if (aState instanceof
        * FinalANDState) { a =
        * StateChartHelpers.convertFromFinalState(aState); }
        * this.getStateChart().setRootNode((CompositeState) a);
        * a.removeIncomingTransition((Transition) ((ArrayList) a
        * .getIncomingTransitions().get(0)); a.setParent(null);
        * a.setIDManual(id); } //
        */
    }
  }
  return result;
}
}
```

C.4.5. EliminateOrStatesWithTwoSubs

Auflistung C.59: Die Klasse `EliminateOrStatesWithTwoSubs`

```
package kiel.optimizer.esterelstudio;

import java.util.ArrayList;

import kiel.dataStructure.ANDState;
import kiel.dataStructure.CompositeState;
import kiel.dataStructure.FinalANDState;
import kiel.dataStructure.FinalORState;
import kiel.dataStructure.Node;
import kiel.dataStructure.NormalTermination;
import kiel.dataStructure.ORState;
import kiel.dataStructure.SimpleState;
import kiel.dataStructure.State;
import kiel.dataStructure.StateChart;
import kiel.dataStructure.WeakAbortion;
import kiel.optimizer.OptimizationRule;
import kiel.optimizer.OptimizerException;
import kiel.util.StateChartHelpers;

/**
 * <p>
 * Eliminates ORStates which may be aborted or have local
 * values, but contains only two substates.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.5 $ last modified $Date: 2006/02/11 11:50:34 $
 */
public final class EliminateOrStatesWithTwoSubs
        extends OptimizationRule {
    /**
     * @param aStateChart aStateChart.
     */
    public EliminateOrStatesWithTwoSubs(final StateChart aStateChart) {
        super(
                aStateChart);
        super.setRuleName("EleminateOrStatesWithTwoSubs");
    }

    /**
     * checks if a optimization for <code>aState</code> is possible.
     *
     * @param aState
     *                a composite state
     * @return true if optimization is possible else false.
     * @see kiel.dataStructure.CompositeState
     */
    public boolean isOptimizationPossible(
            final CompositeState aState) {
        if (StateChartHelpers.getIncommingSuspension(aState).size() > 0) {
            return false;
        }
        if (aState instanceof ORState
                && aState.getSubnodes().size() == 2) {
            State aSubState =
                (State) StateChartHelpers
            .getSubStates(aState).get(0);
             if (aSubState instanceof SimpleState
                        && aState.getParent() != null) {
                ArrayList allOuts =
                    StateChartHelpers.getOutgoingTransitions(aState);
                boolean weak = false;
                boolean normal = false;
                for (int i = 0; i < allOuts.size(); i++) {
                    if (allOuts.get(i) instanceof NormalTermination) {
                        normal = true;
                    }
                    if (allOuts.get(i) instanceof WeakAbortion) {
                        weak = true;
                    }
                }
                if (weak
                        && normal) {
                    return false;
                }
                return true;
            }
            if (aSubState instanceof CompositeState
                    && aState.getParent() == null) {
                if (aSubState instanceof ANDState
                        && (aState.getLocalEvents().size() > 0
                        ||
                        aState.getVariables().size() > 0)) {
                    return false;
                }
            }
                return true;

         }
        return false;
    }

    /**
     * Eleminates macrostates with only two substates.
     * @param aState the macrostate.
     * @param aStateChart the StateChart.
```

```
     * @return true, if something was optimized.
     * @throws OptimizerException if something
     * unexcepted happend.
     */
      public boolean optimizeState(
            final CompositeState aState,
            final StateChart aStateChart)
          throws OptimizerException {
        this.setStateChart(aStateChart);
        if (isOptimizationPossible(aState)) {
          if (aState.getParent() != null) {
          StateChartHelpers
          .deleteOrStateButSaveSubStates((ORState) aState);
          return true;
          }
          State aSubState =
              (State) StateChartHelpers.getSubStates(aState).get(0);
          if (aSubState instanceof FinalANDState || aSubState instanceof FinalORState) {
          aSubState = StateChartHelpers.convertFromFinalState(aSubState);
          while (aState.getSubnodes().size() > 0) {
            aState.removeSubnode(((Node) aState.getSubnodes().get(0)));
          }
          aSubState.addLocalEvents(aState.getLocalEvents());
          aSubState.addVariables(aState.getVariables());
          aStateChart.setRootNode((CompositeState) aSubState);
          aSubState.setParent(null);
          aSubState.setName(aState.getName());
          return true;
          }
        } //if
        return false;
      } //methode
    }
```

C.4.6. EsterelStudioChartOptimizer

Auflistung C.60: Die Klasse `EsterelStudioChartOptimizer`

```
package kiel.optimizer.esterelstudio;

import java.util.ArrayList;
import java.util.Arrays;
import kiel.dataStructure.CompositeState;
import kiel.dataStructure.StateChart;
import kiel.optimizer.OptimizationRule;
import kiel.optimizer.Optimizer;
import kiel.optimizer.OptimizerException;
import kiel.optimizer.OptimizerProperties;
import kiel.util.StateChartHelpers;

/**
 * <p>
 *  This is a main optimizer class..
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 */
public final class EsterelStudioChartOptimizer
       extends Optimizer {
    /**
     * the actual optimized level.
     */
    private int actualLevel = -1;

    /**
     * a flat representation of an statechart sorted by the deepth of a
     * state.
     */
    private StateChartNodeAttributes[] myChart = null;

    /**
     * An array with rules sorted by invokation in a state.
     */
    private OptimizationRule[] theRules = null;

    /**
     * the standard constructor.
     *
     * @throws OptimizerException
     *                 delivers it.
     */
    public EsterelStudioChartOptimizer()
           throws OptimizerException {
        this(
               null);
    }

    /**
     * a constructor which sets the statechart class variable.
     *
     * @param aChart
     *                 a statechart
     * @throws OptimizerException
     *                 delivers Exception
     */
    public EsterelStudioChartOptimizer(final StateChart aChart)
           throws OptimizerException {
        super(
               aChart);
        this.actualLevel = -1;
        this.myChart = null;
        if (aChart != null) {
            this.setmyChart(aChart);
        }
        this.setRules();
    }

    /**
     * starts the optimization of an statechart.
     *
     * @param all
     *                 if true then all levels will be optimized else only the
     *                 lowest unoptimized level .
     * @param aChart
     *                 a statechart
     * @return a statechart
     * @throws OptimizerException
     *                 if something went wrong.
     */
    public StateChart startOptimize(
           final StateChart aChart,
           final boolean all)
           throws OptimizerException {
        boolean aChange = false;
        this.actualLevel = -1;
        this.myChart = null;
        this.setStateChart(null);
        String lastusedRule = "";
        if (aChart != null) {
            if (this.myChart == null) {
                this.setmyChart(aChart);
            }
            long memory=0;
            boolean round = true;
            int deep = -1;
```

```
while (round) {
    int i = getStartIndex();
    if (i > -1) {
        deep = this.myChart[i].getDeep();
        this.actualLevel = this.myChart[i].getDeep();
    }
    aChange = false;
    while (i > -1
           && deep == this.actualLevel) {
        String activerule = "";
        String activeState = "";
        boolean statedeleted = false;
        for (int rulenr = 0;
        rulenr < this.theRules.length && !statedeleted; rulenr++) {
            activerule = this.theRules[rulenr].getRuleName();
            activeState = this.myChart[i].getTheCState()
                   .getName();
            if (OptimizerProperties.getDebugMode()) {
            System.out.println("Before "
                   + activerule +" with "+ activeState
                   + " "
                   + Runtime.getRuntime().freeMemory());
            }
            try {
                if (!(!all && aChange)) {
                    if (OptimizerProperties.getDebugMode()) {
                        //PrettyPrinter.printChart(aChart,
                        // "/home/lku/preRule.txt");
                        ArrayList a = kiel.util.StateChartHelpers
                        .testTransitions(
                            this.myChart[i].getTheCState());
                        if (a.size() > 0) {
                            System.out.println(
                                "Problem before rule : "
                                    + activerule
                                    + " on "
                                    + activeState
                                    + "\n");
                        }
                    }
                    aChange =
                        (this.theRules[rulenr]
                                  .optimizeState(
                                         this.myChart[i]
                                            .getTheCState(),
                        this.getStateChart()));
                    if (aChange) {
                        lastusedRule = activerule;
                        if (lastusedRule
                      .compareTo("EleminateOrStatesWithTwoSubs") == 0
                                  || lastusedRule
                                  .compareTo(
                    "EliminateNotAbortednWithoutLocalsORStates") == 0
                                  || lastusedRule.compareTo(
                                      "UpdateFinalStates") == 0) {
                            statedeleted = true;
                        }
                    }
                    if (OptimizerProperties.getDebugMode()) {
                        System.out.println("After "
                                + activerule +" with "+ activeState
                                + " "
                                + Runtime.getRuntime().freeMemory());
                        if (memory - Runtime.getRuntime().freeMemory() > 10000)
                        {
                            System.out.println("--------------------"+activerule
+"---------------");
                            Runtime.getRuntime().gc();
                            System.out.println("After gc"
                                    + Runtime.getRuntime().freeMemory());
                        }
                        if (aChange) {
                            System.out.println(lastusedRule
                                    + " with "
                                    + activeState);
                            //  PrettyPrinter.printChart(aChart,
                            // "/home/lku/postRule.txt");
                        }
                        ArrayList a =
                            kiel.util.StateChartHelpers
                            .testTransitions(
                                this.myChart[i]
                                        .getTheCState());
                        if (a.size() > 0) {
                            System.out.println(
                                "Problem after rule : "
                                    + lastusedRule
                                    + " on "
                                    + activeState
                                    + "\n");
                        }
                    }
                    if (lastusedRule
                            .compareTo("EliminateDeadStates")
                            == 0) {
                        aChange = false;
                    }
                }
            } catch (Exception e) {
                if (OptimizerProperties.getDebugMode()) {
                    e.printStackTrace();
                }
                throw new OptimizerException(
                        e.getMessage());
            }
        } //for
        i--;
        if (i > -1) {
            deep = this.myChart[i].getDeep();
        } else {
            deep = -1 - 1;
        }
```

```
            } //while
            setmyChart(this.getStateChart());
            if ((all || !aChange)
                   && this.actualLevel > -1
                   && i > -1) {
               round = true;
            } else {
               if (all || !aChange) {
             for (int rulenr = 0; rulenr < this.theRules.length;
               rulenr++) {
                   this.theRules[rulenr]
                               .optimizeState(
                                  this.getStateChart()
                           .getRootNode(),
                        this.getStateChart());
               }
               }
               round = false;
            } // else
         } // while round
         return this.getStateChart();
      } //if
      return null;
   }

   /**
    * @param aState
    *                a state
    * @param deep
    *                the deepth of the states
    * @return all composit states with deep <code>deep</code>
    * @throws OptimizerException
    *                an exception
    */
   private ArrayList getAllCompositeStates(
          final CompositeState aState,
          final int deep)
          throws OptimizerException {
      ArrayList theCStates = new ArrayList();
      if (aState != null) {
          ArrayList allSubStates = StateChartHelpers.getSubStates(aState);
          for (int i = 0; i < allSubStates.size(); i++) {
              if (allSubStates.get(i) instanceof CompositeState) {
                  theCStates.add(new StateChartNodeAttributes(
                         (CompositeState) allSubStates.get(i),
                         deep));
                  ArrayList sub =
                      getAllCompositeStates(
                          (CompositeState) allSubStates.get(i),
                      deep + 1);
                  for (int j = 0; j < sub.size(); j++) {
                      theCStates.add(sub.get(j));
                  }
              } //if
          } //for
      } else {
          throw new OptimizerException(
                 "Found no composite state");
      }
      return theCStates;
   } // getDepth

   /**
    * @return the startindex in <code>myChart</code> of an deeplevel
    */
   private int getStartIndex() {
      int result = -1;
      if (this.myChart.length > 0) {
          if (this.actualLevel == -1) {
              result = this.myChart.length - 1;
          } else {
              int deep = this.myChart[0].getDeep();
              for (int i = 0; i < this.myChart.length
                     && this.actualLevel > deep; i++) {
                 deep = this.myChart[i].getDeep();
                 if (this.actualLevel > deep) {
                     result = i;
                 }
              }
          }
      }
      return result;
   }

   /**
    * @param stateChart
    *                The aStateChart to set.
    * @throws OptimizerException
    *                from called methode.
    */
   private void setmyChart(
          final StateChart stateChart)
          throws OptimizerException {
      this.setStateChart(stateChart);
      //actualLevel = -1;
      if (this.getStateChart() != null) {
          ArrayList allCStates =
              getAllCompositeStates(this.getStateChart()
                  .getRootNode(),
              0);
          this.myChart = new StateChartNodeAttributes[allCStates.size()];
          for (int i = 0; i < this.myChart.length; i++) {
              this.myChart[i] = (StateChartNodeAttributes) allCStates.get(i);
          } //for
          Arrays.sort(this.myChart);
      }
   }

   /**
    * Sets the optimization rules.
    */
   private void setRules() {
```

```
        int rulecounter = 0;
        /*if (OptimizerProperties
            .getOptimizeESTEliminateNeedlessSimpleStates()) {
          rulecounter++;
        }*/
        if (OptimizerProperties
            .getOptimizeESTEliminateNeedLessConditional()) {
          rulecounter++;
          }
        if (OptimizerProperties
            .getOptimizeESTEliminateNeedlessSimpleStates()) {
          rulecounter++;
        }
        if (OptimizerProperties
            .getOptimizeESTJoinFinalStates()) {
          rulecounter++;
        }
        if (OptimizerProperties
            .getOptimizeESTEliminateNeedlessNormalTransitions()) {
          rulecounter++;
        }
        if (OptimizerProperties
            .optimizeESTEliminateNotAbortednWithoutLocalsOrStates()) {
          rulecounter++;
        }
        if (OptimizerProperties.getOptimizeESTOrStatesW2Subs()) {
          rulecounter++;
        }
        if (OptimizerProperties
            .getOptimizeESTUpdateFinalStates()) {
          rulecounter++;
        }
        this.theRules = new OptimizationRule[rulecounter];
        int counter = 0;
        /*if (OptimizerProperties
        * .getOptimizeESTEliminateNeedlessSimpleStates()) {
          this.theRules[counter] = new EliminateNeedlessSimpleStates(
              this.getStateChart());
          counter++;
        }*/
        if (OptimizerProperties.getOptimizeESTEliminateNeedLessConditional()) {
          this.theRules[counter] = new EliminateNeedlessConditional(
              this.getStateChart());
          counter++;
        }
        if (OptimizerProperties
            .getOptimizeESTEliminateNeedlessSimpleStates()) {
          this.theRules[counter] =
              new EliminateNeedlessSimpleStates(
                  this.getStateChart());
          counter++;
        }
        if (OptimizerProperties.getOptimizeESTJoinFinalStates()) {
          this.theRules[counter] = new JoinFinalSimpleStates(
              this.getStateChart());
          counter++;
        }
        if (OptimizerProperties
            .getOptimizeESTEliminateNeedlessNormalTransitions()) {
          this.theRules[counter] =
              new EliminateNeedlessNormalTerminations(
                  this.getStateChart());
          counter++;
        }
        if (OptimizerProperties
            .optimizeESTEliminateNotAbortednWithoutLocalsOrStates()) {
          this.theRules[counter] =
              new EliminateNotAbortednWithoutLocalsORStates(
                  this.getStateChart());
          counter++;
        }
        if (OptimizerProperties.getOptimizeESTOrStatesW2Subs()) {
          this.theRules[counter] = new EliminateOrStatesWithTwoSubs(
              this.getStateChart());
          counter++;
        }
        if (OptimizerProperties.getOptimizeESTUpdateFinalStates()) {
          this.theRules[counter] = new UpdateFinalStates(
              this.getStateChart());
          counter++;
        }
      }
    }
```

C.4.7. JoinFinalSimpleStates

Auflistung C.61: Die Klasse `JoinFinalSimpleStates`

```
package kiel.optimizer.esterelstudio;

import java.util.ArrayList;
import kiel.dataStructure.ANDState;
import kiel.dataStructure.CompositeState;
import kiel.dataStructure.FinalSimpleState;
import kiel.dataStructure.Node;
import kiel.dataStructure.StateChart;
import kiel.dataStructure.Transition;
import kiel.optimizer.OptimizationRule;
import kiel.optimizer.OptimizerException;
import kiel.util.StateChartHelpers;

/**
 * <p>
 *  Joins FinalSimpleStates in a macrostate.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.18 $ last modified $Date: 2006/02/06 18:36:04 $
 */
public class JoinFinalSimpleStates extends OptimizationRule {
    /**
     *
     * @param aChart aStateChart.
     */
    public JoinFinalSimpleStates(final StateChart aChart) {
        super(aChart);
        super.setRuleName("JoinFinalStates");
    }

    /**
     * checks if a optimization for <code>aState</code> is possible.
     *
     * @param aState
     *                a composite state
     * @return true if optimization is possible else false.
     * @see kiel.dataStructure.CompositeState
     */
    public final boolean isOptimizationPossible(final CompositeState aState) {
        if (aState instanceof ANDState) {
            return false;
        }
        return true;
    }

    /**
     * optimization for <code>aState</code>.
     *
     * @param aState
     *                a composite state
     * @param aStateChart a statechart
     * @return null
     * @throws OptimizerException
     *                not throw, may be trown in called methods.
     * @see kiel.dataStructure.CompositeState
     */
    public final boolean optimizeState(final CompositeState aState,
            final StateChart aStateChart)
            throws OptimizerException {
        boolean result = false;
        this.setStateChart(aStateChart);
        if (isOptimizationPossible(aState)) {
            // if there is more than one final simple state. Join them
            ArrayList allFinalSimpleStates = StateChartHelpers
                    .getSubFinalSimpleStates(aState);
            if (allFinalSimpleStates.size() > 1) {
                result = true;
                FinalSimpleState theOne = (FinalSimpleState)
                allFinalSimpleStates.get(0);
                for (int i = 1; i < allFinalSimpleStates.size(); i++) {
                    ArrayList allTransToFSState = StateChartHelpers
                            .getIncommingTransitions((Node) allFinalSimpleStates
                                    .get(i));
                    while (allTransToFSState.size() > 0) {
                        ((Transition) allTransToFSState.get(0))
                                .setTarget(theOne);
                        ((Node) allFinalSimpleStates.get(i))
                                .removeIncomingTransition(
                                        ((Transition) allTransToFSState
                                        .get(0)));
                    }
                    result = true;
                    aState.removeSubnode(((Node) allFinalSimpleStates.get(i)));
                }
            }
        }
        return result;
    }
}
```

C.4.8. StateChartNodeAttributes

Auflistung C.62: Die Klasse `StateChartNodeAttributes`

```
package kiel.optimizer.esterelstudio;

import java.util.ArrayList;
import kiel.dataStructure.CompositeState;

/**
 * <p>
 *  A class for get an flat struckture from a statechart
 * where the Composite states know their depth.
 *
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.22 $ last modified $Date: 2006/02/06 18:36:04 $
 */
public class StateChartNodeAttributes implements Comparable {
/**
 * a CompositeState.
 */
   private CompositeState theCState;
/**
 * Depth of <code>theCState</code>.
 */
   private int deep;
/**
 * Names of the used optimizations on the
 * <code>theCState</code>.
 */
   private ArrayList usedOptimaziations = null;

   /**
    *A constructor which sets the class variables
    *theCState and deep.
    *@param aCState a composite state.
    *@param depthInStateChart the depth of aCState
    *in a StateChart.
    */
   public StateChartNodeAttributes(final CompositeState aCState,
          final int depthInStateChart) {
      super();
      this.theCState = aCState;
      this.deep = depthInStateChart;
      this.usedOptimaziations = new ArrayList();
   }

/**
 * @return Returns the deep.
 */
public final int getDeep() {
   return this.deep;
}
/**
 * @param adeep The deep to set.
 */
public final void setDeep(final int adeep) {
   this.deep = adeep;
}
/**
 * @return Returns the theCState.
 */
public final CompositeState getTheCState() {
   return this.theCState;
}
/**
 * @param aCState The theCState to set.
 */
public final void setTheCState(final CompositeState aCState) {
   this.theCState = aCState;
}
/**
 * @return Returns the usedOptimaziations.
 */
public final ArrayList getUsedOptimaziations() {
   return this.usedOptimaziations;
}
/**
 * @param anUsedOptimaziations The usedOptimaziations to set.
 */
public final void setUsedOptimaziations(
      final ArrayList anUsedOptimaziations) {
   this.usedOptimaziations = anUsedOptimaziations;
}
/**
 * adds an rule name to usedOptimizations.
 * @param ruleName a rule name.
 */
public final void addOptimizationRule(
      final String ruleName) {
   this.usedOptimaziations.add(ruleName);
}
/**
 * compares the object with arg0.
 * @param arg0 a object.
 * @return 0 if equal , -1 if less else +1.
 */
public final int compareTo(final Object arg0) {
   int result = 0;
```

```
    }
    if (arg0 instanceof StateChartNodeAttributes) {
        if (this.deep < ((StateChartNodeAttributes) arg0).deep) {
            result = -1;
        } else if (this.deep > ((StateChartNodeAttributes) arg0).deep) {
            result = 1;
        } else if (this.deep == ((StateChartNodeAttributes) arg0).deep) {
            result = 0;
        }
    } else {
        result = 1;
    }
    return result;
}

/**
 * @return a String
 */
public final String toString() {
    return this.theCState.getID()
        + " :\n  "
        + this.deep
        + "\n";
    }
}
```

C.4.9. UpdateFinalStates

Auflistung C.63: Die Klasse `UpdateFinalStates`

```
package kiel.optimizer.esterelstudio;

import java.util.ArrayList;

import kiel.dataStructure.ANDState;
import kiel.dataStructure.CompositeState;
import kiel.dataStructure.FinalANDState;
import kiel.dataStructure.FinalORState;
import kiel.dataStructure.StateChart;
import kiel.optimizer.OptimizationRule;
import kiel.optimizer.OptimizerException;
import kiel.util.StateChartHelpers;

/**
 * <p>
 *  Converts final states
 * which have no final substates to non final states.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 *
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * @version $Revision: 1.4 $ last modified $Date: 2006/02/17 20:00:22 $
 */
public class UpdateFinalStates
        extends OptimizationRule {
    /**
     * @param aChart
     *                aStateChart.
     */
    public UpdateFinalStates(final StateChart aChart) {
        super(
                aChart);
        super.setRuleName("UpdateFinalStates");
    }

    /**
     * checks if a optimization for <code>aState</code> is possible.
     *
     * @param aState
     *                a composite state
     * @return true if optimization is possible else false.
     * @see kiel.dataStructure.CompositeState
     */
    public final boolean isOptimizationPossible(
            final CompositeState aState) {
        if (aState instanceof FinalORState && aState.getParent() != null) {
            if (StateChartHelpers.getSubFinalStates(aState).size() == 0) {
                return true;
                }
        } else if (aState instanceof FinalANDState && aState.getParent() != null) {
            ArrayList allRegions = ((ANDState) aState).getSubnodes();
            for (int i = 0; i < allRegions.size(); i++) {
                if (StateChartHelpers
                        .getSubFinalStates(
                                (CompositeState) allRegions
                                .get(i))
                                .size() == 0) {
                    return true;
                }
            }
        }
        return false;
    }

    /**
     * optimization for <code>aState</code>.
     *
     * @param aState
     *                a composite state
     * @param aStateChart
     *                a statechart
     * @return true if something happedl
     * @throws OptimizerException
     *                not throw, may be trown in called methods.
     * @see kiel.dataStructure.CompositeState
     */
    public final boolean optimizeState(
            final CompositeState aState,
            final StateChart aStateChart)
            throws OptimizerException {
        boolean result = false;
        this.setStateChart(aStateChart);
        if (isOptimizationPossible(aState)) {
            // if there is more than one final simple state.
                StateChartHelpers.convertFromFinalState(aState);
                result = true;
        }
        return result;
    }
}
```