

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

1. Einleitung	21
2. Grundlagen	25
2.1. Vorgaben	25
2.2. Esterel	26
2.2.1. Deklarationen	27
2.2.2. Ausdrücke	29
2.3. Einführung in die Syntax der SyncChart	31
2.3.1. Zustände	31
2.3.2. Transitionen	32
2.4. Verwendete Programme	35
2.4.1. Der <i>Columbia Esterel Compiler</i>	35
2.4.2. <i>KIEL</i>	35
3. Transformation von Esterel in SyncCharts	39
3.1. Einführungen	39
3.2. Die Transformation der Esterel-Deklarationen	39
3.3. Die Transformation der Esterel-Ausdrücke	39
3.4. Die Transformation der Esterel-Befehle	40
3.4.1. Die Transformationsgrammatik	46
3.4.2. Das Modul	47
3.4.3. Der Befehl nothing	48
3.4.4. Der Befehl halt	48
3.4.5. Der Befehl pause	49
3.4.6. Der Befehl abort	50
3.4.7. Der Befehl assign	52
3.4.8. Der Befehl await	52
3.4.9. Der Befehl doupto	54
3.4.10. Der Befehl dowatching	55
3.4.11. Der Befehl emit	57
3.4.12. Der Befehl every	57
3.4.13. Der Befehl if	59
3.4.14. Der Befehl localsignal	60
3.4.15. Der Befehl localvariable	61
3.4.16. Der Befehl loop	62
3.4.17. Der Befehl loopeach	63

Inhaltsverzeichnis

3.4.18. Der Befehl parallel	65
3.4.19. Der Befehl present	66
3.4.20. Der Befehl call	67
3.4.21. Der Befehl sequence	68
3.4.22. Der Befehl suspend	69
3.4.23. Der Befehl sustain	70
3.4.24. Der Befehl trap	75
3.4.25. Der Befehl exit	84
3.4.26. Der Befehl weakabort	85
3.5. Nicht transformierbare Befehle	86
4. Optimierung von SyncCharts	89
4.0.1. Entfernen unnötiger <i>Conditional</i> -Pseudozustände	89
4.0.2. Entfernen unnötiger, einfacher Zustände	90
4.0.3. Zusammenfassen einfacher, finaler Zustände	91
4.0.4. Entfernen unnötiger <i>normal terminations</i>	92
4.0.5. Entfernen unnötiger Makrozustände	92
4.0.6. Entfernen von Makrozuständen mit nur zwei Unterzuständen	93
4.0.7. Überprüfen auf finalen Charakter	93
4.1. Der Algorithmus zum Optimieren eines Makrozustandes	94
5. Transformation am Beispiel von ABRO	97
5.1. Beispielhafte Transformation von Esterel nach SyncChart	97
5.2. Optimierung	102
6. Die Implementation	107
6.0.1. Die Verwendung des cec	107
6.0.2. <i>XPath</i>	107
6.1. Die Java-Klassenstruktur für Esterel	108
6.1.1. Die Superklassen	109
6.1.2. Die Klasse <i>EsterelModule</i>	112
6.2. <i>Parsen</i> eines Esterel Moduls	112
6.3. Die Transformation in die graphische Darstellung	113
6.3.1. Der Transformationsalgorithmus	113
6.3.2. Kompromisse bei der Transformation in <i>KIEL</i>	114
6.4. Optimierung	114
7. Zusammenfassung und Ausblick	117
7.1. Zusammenfassung	117
7.2. Ausblick	117
A. Literaturverzeichnis	119

B. Bedienungsanleitung	123
B.1. Transformation von Esterel nach SyncChart	123
B.1.1. Die Datei <code>esterel.properties</code>	123
B.1.2. Die Datei <code>esterel2estudio.properties</code>	124
B.2. Der Optimizer	124
C. Java Code für die Transformation von Esterel in SyncCharts	127
C.1. Paket <code>kiel.fileInterface.estrel</code>	127
C.1.1. <code>EsterelFileFilter</code>	128
C.1.2. <code>Esterel</code>	129
C.1.3. <code>EsterelParserException</code>	134
C.1.4. <code>EsterelParser</code>	135
C.1.5. <code>EsterelProperties</code>	139
C.2. Paket <code>kiel.fileInterface.estrel.estrel2estudio</code>	142
C.2.1. <code>AbortEsterelStatement</code>	143
C.2.2. <code>AssignEsterelStatement</code>	148
C.2.3. <code>AwaitEsterelStatement</code>	151
C.2.4. <code>DoUpToEsterelStatement</code>	155
C.2.5. <code>DoWatchingEsterelStatement</code>	158
C.2.6. <code>EmitEsterelStatement</code>	161
C.2.7. <code>Esterel2EstudioException</code>	164
C.2.8. <code>Esterel2EstudioProperties</code>	165
C.2.9. <code>EsterelConstant</code>	167
C.2.10. <code>EsterelDeclaration</code>	171
C.2.11. <code>EsterelDeclarationSortByID</code>	173
C.2.12. <code>EsterelDeclarationSortByName</code>	174
C.2.13. <code>EsterelDelayExpression</code>	175
C.2.14. <code>EsterelExpression</code>	179
C.2.15. <code>EsterelFunctionDeclaration</code>	187
C.2.16. <code>EsterelFunction</code>	190
C.2.17. <code>EsterelModule</code>	192
C.2.18. <code>EsterelProcedureDeclaration</code>	201
C.2.19. <code>EsterelRelation</code>	204
C.2.20. <code>EsterelSignalExpression</code>	207
C.2.21. <code>EsterelSignal</code>	212
C.2.22. <code>EsterelStatement</code>	218
C.2.23. <code>EsterelTaskDeclaration</code>	221
C.2.24. <code>EsterelTypeDeclaration</code>	224
C.2.25. <code>EsterelVariable</code>	226
C.2.26. <code>EveryEsterelStatement</code>	231
C.2.27. <code>ExitEsterelStatement</code>	234
C.2.28. <code>HaltEsterelStatement</code>	237
C.2.29. <code>IfEsterelStatement</code>	239
C.2.30. <code>LocalSignalDeclarationEsterelStatement</code>	243

Inhaltsverzeichnis

C.2.31.	LocalVariableEsterelStatement	246
C.2.32.	LoopEachEsterelStatement	249
C.2.33.	LoopEsterelStatement	252
C.2.34.	NothingEsterelStatement	255
C.2.35.	ParallelEsterelStatement	257
C.2.36.	PauseEsterelStatement	260
C.2.37.	PresentThenElse	262
C.2.38.	ProcedureCallEsterelStatement	266
C.2.39.	RepeatEsterelStatement	269
C.2.40.	SequenceEsterelStatement	274
C.2.41.	SuspendEsterelStatement	277
C.2.42.	SustainEsterelStatement	280
C.2.43.	TaskCallEsterelStatement	284
C.2.44.	TrapEsterelStatement	288
C.3.	Paket <code>kiel.optimizer</code>	295
C.3.1.	OptimizationRule	296
C.3.2.	OptimizerChooser	297
C.3.3.	OptimizerException	298
C.3.4.	Optimizer	299
C.3.5.	OptimizerProperties	300
C.4.	Paket <code>kiel.optimizer.estrelstudio</code>	303
C.4.1.	EliminateNeedlessConditional	304
C.4.2.	EliminateNeedlessNormalTerminations	307
C.4.3.	EliminateNeedlessSimpleStates	308
C.4.4.	EliminateNotAbortednWithoutLocalsORStates	311
C.4.5.	EliminateOrStatesWithTwoSubs	313
C.4.6.	EsterelStudioChartOptimizer	315
C.4.7.	JoinFinalSimpleStates	319
C.4.8.	StateChartDataAttributes	320
C.4.9.	UpdateFinalStates	322

Abbildungsverzeichnis

1.1. Lochstreifen Programm	21
1.2. Die Gegenüberstellung von Esterel Programmcode und einem entsprechenden SyncChart	24
2.1. Die Struktur von <i>KIEL</i>	36
3.1. Die Alternative zu <code>doupto</code>	54
3.2. Die Alternative zu <code>dowatching</code>	56
3.3. Alternative zu <code>every</code>	58
3.4. Alternative zu <code>loopeach</code>	64
3.5. Alternative zu <code>sustain</code>	71
6.1. UML-Darstellung der Klasse <code>EsterelDeclaration</code>	109
6.2. UML-Darstellung der Klasse <code>EsterelStatement</code>	109
6.3. UML-Darstellung der Klasse <code>EsterelModule</code>	111
C.1. Klassendiagramm <code>AbortEsterelStatement</code>	143
C.2. Klassendiagramm <code>AssignEsterelStatement</code>	148
C.3. Klassendiagramm <code>AwaitEsterelStatement</code>	151
C.4. Klassendiagramm <code>DoUpToEsterelStatement</code>	155
C.5. Klassendiagramm <code>DoWatchingEsterelStatement</code>	158
C.6. Klassendiagramm <code>EmitEsterelStatement</code>	161
C.7. Klassendiagramm <code>EsterelConstant</code>	167
C.8. Klassendiagramm <code>EsterelDeclaration</code>	171
C.9. Klassendiagramm <code>EsterelDelayExpression</code>	175
C.10. Klassendiagramm <code>EsterelExpression</code>	179
C.11. Klassendiagramm <code>EsterelFunctionDeclaration</code>	187
C.12. Klassendiagramm <code>EsterelModule</code>	200
C.13. Klassendiagramm <code>EsterelProcedureDeclaration</code>	201
C.14. Klassendiagramm <code>EsterelRelation</code>	204
C.15. Klassendiagramm <code>EsterelSignalExpression</code>	207
C.16. Klassendiagramm <code>EsterelSignal</code>	212
C.17. Klassendiagramm <code>EsterelStatement</code>	218
C.18. Klassendiagramm <code>EsterelTaskDeclaration</code>	221
C.19. Klassendiagramm <code>EsterelTypeDeclaration</code>	224
C.20. Klassendiagramm <code>EsterelVariable</code>	226
C.21. Klassendiagramm <code>EveryEsterelStatement</code>	231

Abbildungsverzeichnis

C.22.Klassediagramm ExitEsterelStatement	234
C.23.Klassediagramm HaltEsterelStatement	237
C.24.Klassediagramm IfEsterelStatement	239
C.25.Klassediagramm LoopEachEsterelStatement	249
C.26.Klassediagramm LoopEsterelStatement	252
C.27.Klassediagramm NothingEsterelStatement	255
C.28.Klassediagramm ParallelEsterelStatement	257
C.29.Klassediagramm PauseEsterelStatement	260
C.30.Klassediagramm ProcedureCallEsterelStatement	266
C.31.Klassediagramm RepeatEsterelStatement	269
C.32.Klassediagramm SequenceEsterelStatement	274
C.33.Klassediagramm SuspendEsterelStatement	277
C.34.Klassediagramm SustainEsterelStatement	280
C.35.Klassediagramm TaskCallEsterelStatement	284
C.36.Klassediagramm TrapEsterelStatement	288
C.37.Klassediagramm OptimizerRules	303
C.38.Klassediagramm EsterelStudioChartOptimizer	303

Verzeichnis der Auflistungen

1.1. Maschinensprache	22
1.2. Summe in Pascal	23
1.3. ABRO	24
C.1. Die Klasse <code>EsterelFileFilter</code>	128
C.2. Die Klasse <code>Esterel</code>	129
C.3. Die Klasse <code>EsterelParserException</code>	134
C.4. Die Klasse <code>EsterelParser</code>	135
C.5. Die Klasse <code>EsterelProperties</code>	139
C.6. Die Klasse <code>AbortEsterelStatement</code>	143
C.7. Die Klasse <code>AssignEsterelStatement</code>	148
C.8. Die Klasse <code>AwaitEsterelStatement</code>	151
C.9. Die Klasse <code>DoUpToEsterelStatement</code>	155
C.10. Die Klasse <code>DoWatchingEsterelStatement</code>	158
C.11. Die Klasse <code>EmitEsterelStatement</code>	161
C.12. Die Klasse <code>Esterel2EstudioException</code>	164
C.13. Die Klasse <code>Esterel2EstudioProperties</code>	165
C.14. Die Klasse <code>EsterelConstant</code>	167
C.15. Die Klasse <code>EsterelDeclaration</code>	171
C.16. Die Klasse <code>EsterelDeclarationSortByID</code>	173
C.17. Die Klasse <code>EsterelDeclarationSortByName</code>	174
C.18. Die Klasse <code>EsterelDelayExpression</code>	175
C.19. Die Klasse <code>EsterelExpression</code>	179
C.20. Die Klasse <code>EsterelFunctionDeclaration</code>	187
C.21. Die Klasse <code>EsterelFunction</code>	190
C.22. Die Klasse <code>EsterelModule</code>	192
C.23. Die Klasse <code>EsterelProcedureDeclaration</code>	201
C.24. Die Klasse <code>EsterelRelation</code>	204
C.25. Die Klasse <code>EsterelSignalExpression</code>	207
C.26. Die Klasse <code>EsterelSignal</code>	212
C.27. Die Klasse <code>EsterelStatement</code>	218
C.28. Die Klasse <code>EsterelTaskDeclaration</code>	221
C.29. Die Klasse <code>EsterelTypeDeclaration</code>	224
C.30. Die Klasse <code>EsterelVariable</code>	226
C.31. Die Klasse <code>EveryEsterelStatement</code>	231
C.32. Die Klasse <code>ExitEsterelStatement</code>	234
C.33. Die Klasse <code>HaltEsterelStatement</code>	237
C.34. Die Klasse <code>IfEsterelStatement</code>	239

Verzeichnis der Auflistungen

C.35.Die Klasse LocalSignalDeclarationEsterelStatement	243
C.36.Die Klasse LocalVariableEsterelStatement	246
C.37.Die Klasse LoopEachEsterelStatement	249
C.38.Die Klasse LoopEsterelStatement	252
C.39.Die Klasse NothingEsterelStatement	255
C.40.Die Klasse ParallelEsterelStatement	257
C.41.Die Klasse PauseEsterelStatement	260
C.42.Die Klasse PresentThenElse	262
C.43.Die Klasse ProcedureCallEsterelStatement	266
C.44.Die Klasse RepeatEsterelStatement	269
C.45.Die Klasse SequenceEsterelStatement	274
C.46.Die Klasse SuspendEsterelStatement	277
C.47.Die Klasse SustainEsterelStatement	280
C.48.Die Klasse TaskCallEsterelStatement	284
C.49.Die Klasse TrapEsterelStatement	288
C.50.Die Klasse OptimizationRule	296
C.51.Die Klasse OptimizerChooser	297
C.52.Die Klasse OptimizerException	298
C.53.Die Klasse Optimizer	299
C.54.Die Klasse OptimizerProperties	300
C.55.Die Klasse EliminateNeedlessConditional	304
C.56.Die Klasse EliminateNeedlessNormalTerminations	307
C.57.Die Klasse EliminateNeedlessSimpleStates	308
C.58.Die Klasse EliminateNotAbortednWithoutLocalsORStates	311
C.59.Die Klasse EliminateOrStatesWithTwoSubs	313
C.60.Die Klasse EsterelStudioChartOptimizer	315
C.61.Die Klasse JoinFinalSimpleStates	319
C.62.Die Klasse StateChartDataAttributes	320
C.63.Die Klasse UpdateFinalStates	322

Verzeichnis der Akronyme

KIEL Kiel Integrated Environment for Layout

CEC Columbia Esterel Compiler

EBNF Erweiterte Backus-Naur-Form

XML Extended Markup Language

UML Unified Modelling Language

Verzeichnis der Auflistungen

Verzeichnis der Transformationsregeln

1.	module	47
2.	nothing	48
3.	halt	49
4.	pause	50
5.	abort	51
6.	abort in weak abort	51
7.	assign	52
8.	await	53
9.	await in await case	53
10.	doupto	54
11.	dowatching	56
12.	emit	57
13.	every	58
14.	if	60
15.	localsignal	61
16.	localvariable	62
17.	loop	63
18.	loopeach	64
19.	parallel	65
20.	present	67
21.	then in case	67
22.	call	68
23.	sequence	69
24.	suspend	70
25.	sustain	71
26.	trap	76
27.	exit	84
28.	weakabort	86
29.	weak abort in weak abort case	86

Verzeichnis der Auflistungen

Verzeichnis der Optimierungsregeln

1.	Entfernen unnötiger <i>Conditional</i> -Pseudozustände	89
2.	Entfernen unnötiger, einfacher Zustände 1	90
3.	Entfernen unnötiger, einfacher Zustände 2	91
4.	Zusammenfassen einfacher, finaler Zustände	91
5.	Entfernen unnötiger <i>normal terminations</i>	92
6.	Entfernen unnötiger Makrozustände	93
7.	Entfernen von Makrozuständen mit nur zwei Unterzuständen	93
8.	Überprüfen auf finalen Charakter	94

Verzeichnis der Auflistungen

1. Einleitung

Wir leben in einer Zeit, die durch die Verbesserungen in der Chip-technologie 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, selbst 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-

1. Einleitung

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 ermöglichen es nun, Programme in einer verständlicheren Form zu erstellen. Intern wurden die Befehle trotzdem 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 dargestellte 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 Abbildung 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.

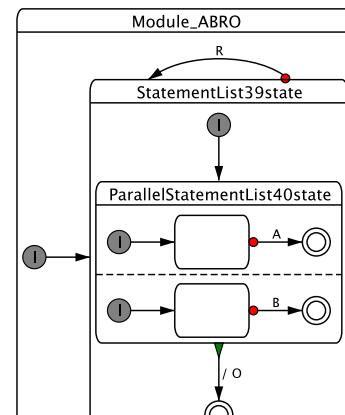
1. Einleitung

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 Esterel-Studio 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].

2. Grundlagen

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. Des Weiteren 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 wird 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 wird. Als letztes folgt dann der Anweisungsteil, der aus mindestens einem Esterel-Befehl besteht. Die einzelnen Esterel-Befehle 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 Signallamen im aufgerufenen Modul textlich durch die angegebenen, bereits vorhandenen Signallamen 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.

Des Weiteren 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 verhalten 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 $\langle DeclarationList \rangle$ in einem Esterel Module ist folgendermaßen aufgebaut.

```

 $\langle Ziffer \rangle ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'$ 
 $\langle UnsignedIntegerConstant \rangle ::= <Ziffer> \{ <Ziffer> \}$ 
 $\langle Letter \rangle ::= 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' | 'k' | 'l' | 'm'$ 
 $\quad | 'n' | 'o' | 'p' | 'q' | 'r' | 's' | 't' | 'v' | 'w' | 'x' | 'y' | 'z'$ 
 $\quad | '_' | 'A' | 'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H' | 'I' | 'J' | 'K'$ 
 $\quad | 'L' | 'M' | 'N' | 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' | 'V' | 'W'$ 
 $\quad | 'X' | 'Y' | 'Z'$ 
 $\langle Identifier \rangle ::= <Letter> \{ (<Letter>) | <Ziffer> \}$ 
 $\langle TypeIdentifier \rangle ::= <Identifier>$ 
 $\langle FunctionIdentifier \rangle ::= <Identifier>$ 
 $\langle ProcedureIdentifier \rangle ::= <Identifier>$ 
 $\langle TaskIdentifier \rangle ::= <Identifier>$ 
 $\langle SignalIdentifier \rangle ::= <Identifier>$ 
 $\langle VariableIdentifier \rangle ::= <Identifier>$ 
 $\langle ConstantIdentifier \rangle ::= <Identifier>$ 
 $\langle StringConstant \rangle ::= <Identifier>$ 

```

2. Grundlagen

```

⟨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’ (FunctionI-
dentifer | ‘+’ | ‘*’ | ‘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>
| <VariablenIdentifier>

```

2. Grundlagen

```

⟨Constant⟩ ::= <ConstantLiteral>
| <UnsignedNumber>

⟨ConstantLiteral⟩ ::= <ConstantIdentifier>
| ‘true’
| ‘false’
| <StringConstant>

⟨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.

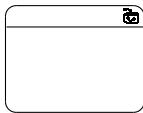
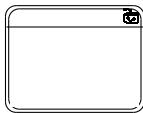
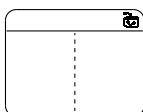
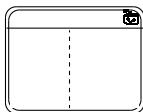
2.3.1. Zustände

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

Des Weiteren existieren *Makrozustände*, die wiederum ein Netz aus Zuständen und Transitionen beinhalten können. Die Makrozustände bilden so eine Hierarchie 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.

- ① Ein Initialer Zustand ist ein Pseudozustand, der
 - : den Startpunkt innerhalb des Makrozustandes darstellt.

2. Grundlagen

- : 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 Hierarchie 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 verschiedenen 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 Bedingungen ein Zustand über diese Transition verlassen werden soll.

2.3. Einführung in die Syntax der SyncChart

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.

$$\langle TransitionLabel \rangle ::= (\# \mid \langle factor \rangle) \langle trigger \rangle \{ \langle condition \rangle \} \{/ \langle effect \rangle}$$

<factor> ist ein ganzzahliger Ausdruck, der dafür sorgt, dass eine Transition erst nach $\langle factor \rangle$ -maligem Auftreten des $\langle trigger \rangle$ ausgelöst wird.

'#' steht für sofortige erstmalige Überprüfung des $\langle triggers \rangle$.

<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 Transitionsbeschriftungen beschrieben 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.

2. Grundlagen

Die Transitionen sind.

Die *Conditional*-Transition ist optisch nur durch den Startzustand von der *weak abortions* und der Initialen Transition zu 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*-Transition notwendig 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 Hauptteil 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 nach sich ziehen.

2.4.2. *KIEL*

KIEL steht für Kiel Integrated Environment for Layout und ist eine Entwicklung des Lehrstuhls für Echtzeitsysteme und Egebiete 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 Entwicklung von *KIEL* möglich. Im Folgenden wird kurz auf die einzelnen Module eingegangen.

2. Grundlagen

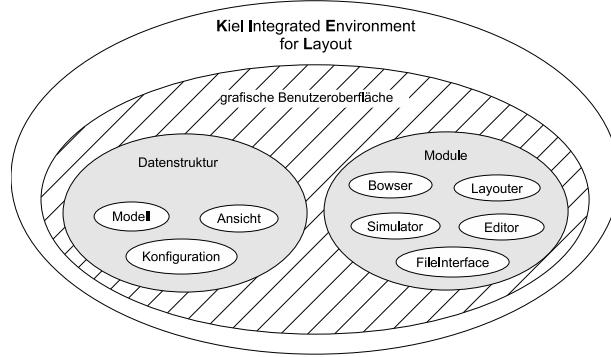


Abbildung 2.1.: Die Struktur von *KIEL*

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. Des Weiteren 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 verschiedenen Statechart-Dialekte in *KIEL*. Die Umwandlung 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.

2. Grundlagen

3. Transformation von Esterel in SyncCharts

3.1. Einführungen

Bevor die einzelnen 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 *Statements* 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 nicht-terminaler 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 $k \geq 2$
	$\uparrow p$
suspend p when immediate SE end	SE· ▷p
await immediate SE do p end	SE· ⇒p
await SE do p end	SE⇒p
weakly abort p when immediate	SE· >p
weakly abort p when	SE>p
abort p when immediate	SE· ≫p
abort p when SE	SE≫p
	$\equiv \{(SE?1,2)^*\};SE▷p$
	$\equiv \{(SE(\uparrow p;2),1)^*\}$
	$\equiv 1;SE\cdot \Rightarrow p$
	$\equiv \{(\uparrow p;2) s \cdot \Rightarrow\}$
	$\equiv \{(\uparrow p;2) s \Rightarrow\}$
	$\equiv SE\cdot >SE(SE\cdot \triangleright p)$
	$\equiv SE>SE(SE\triangleright p)$

3.4. Die Transformation der Esterel-Befehle

Die folgenden Schreibweisen dienen zur Ergänzung, sei dazu $x \in \mathbb{N}$.

Definition 6

```
present case SE1 do p1
... case SEn do pn      SE1, ..., SEn?p1, ..., pn
else q end
```

Definition 7

await x SE do p x SE --> p Dazu gelte die Ersetzung

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

Definition 8

await DE do p DE~>p Es gelte die Ersetzung

$$DE \sim p \equiv \begin{cases} xSE \rightarrow p & , \text{ falls } DE = \langle \text{factor} \rangle SE \\ SE \cdot \Rightarrow p & , \text{ falls } DE = \text{immediate SE} \\ SE \Rightarrow p & , \text{ sonst} \end{cases} .$$

Definition 9

```
await case DE1 do p1
... case DEn do pn      DE1, ..., DEn?p1, ..., pn
end
```

Definition 10

```
weak abort case   DE1
do p1
... case DEn do pn      (\{\(\uparrow p;2)|DE1, ..., DEn?!(H1);2, ..., (Hn);2\};
end
                                         (H1), ..., (Hn)?p1, ..., pn) \ (H1) ... \ (Hn)
```

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

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

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

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

3. Transformation von Esterel in SyncCharts

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

3.4. Die Transformation der Esterel-Befehle

$$\begin{array}{l}
(\text{seq1}) \quad \frac{\text{p} \xrightarrow[E]{E_p,k} \text{p}', k \neq 0}{\text{p}; \text{q} \xrightarrow[E]{E_p,k} \text{p}', ; \text{q}} \\
\\
(\text{seq2}) \quad \frac{\text{p} \xrightarrow[E]{E_p,0} \text{p}', \quad \text{q} \xrightarrow[E]{E_q,k} \text{q},}{\text{p}; \text{q} \xrightarrow[E]{E_p \cup E_q, k} \text{q},} \\
\\
(\text{loop}) \quad \frac{\text{p} \xrightarrow[E]{E_p,k} \text{p}', k \neq 0}{\text{p}^* \xrightarrow[E]{E_p,k} \text{p}', ; \text{p}^*} \\
\\
(\text{parallel}) \quad \frac{\text{p} \xrightarrow[E]{E_p,k} \text{p}', \quad \text{q} \xrightarrow[E]{E_q,l} \text{q},}{\text{p} \mid \text{q} \xrightarrow[E]{E_p \cup E_q, \max(k, l)} \text{p}, \mid \text{q},} \\
\\
(\text{present+}) \quad \frac{\text{SE}^+ \in E \quad \text{p} \xrightarrow[E]{E_p,k} \text{p}'}{\text{SE}? \text{p}, \text{q} \xrightarrow[E]{E_p,k} \text{p}'} \\
\\
(\text{present-}) \quad \frac{\text{SE}^- \in E \quad \text{q} \xrightarrow[E]{E_q,k} \text{q},}{\text{SE}? \text{p}, \text{q} \xrightarrow[E]{E_p,k} \text{q},} \\
\\
(\text{sig+}) \quad \frac{\text{p} \xrightarrow[E*\mathcal{S}^+,k]{} \text{p}', \quad \mathcal{S}(E_p) = \mathcal{S}(E) \setminus S}{\text{p} \setminus \mathcal{S} \xrightarrow[E]{E_p,k} \text{p}', \setminus \mathcal{S}} \\
\\
(\text{sig-}) \quad \frac{\text{p} \xrightarrow[E*\mathcal{S}^-,k]{} \text{p}', \quad \mathcal{S}(E_p) = \mathcal{S}(E) \setminus S}{\text{p} \setminus \mathcal{S} \xrightarrow[E]{E_p,k} \text{p}', \setminus \mathcal{S}} \\
\\
(\text{exit}) \quad k \xrightarrow[E]{\emptyset, k} 0 \\
\\
(\text{trap1}) \quad \frac{\text{p} \xrightarrow[E]{E_p,k} \text{p}', k = 0 \vee k = 2}{\{\text{p}\} \xrightarrow[E]{E_p,0} 0} \\
\\
(\text{trap2}) \quad \frac{\text{p} \xrightarrow[E]{E_p,k} \text{p}', k = 1 \vee k > 2}{\{\text{p}\} \xrightarrow[E]{E_p, \downarrow k} \{\text{p}'\}}
\end{array}$$

3. Transformation von Esterel in SyncCharts

$$\begin{array}{l}
 (\text{shift}) \frac{\begin{array}{c} p \xrightarrow{E_p,k} p, \\ \uparrow p \end{array}}{\uparrow p \xrightarrow[E]{E_p,\uparrow k} \uparrow p,} \\
 (\text{suspend1}) \frac{\begin{array}{c} p \xrightarrow{E_p,0} p, \\ \text{SE} \supset p \end{array}}{\text{SE} \supset p \xrightarrow[E]{E_p,0} p,} \\
 (\text{suspend2}) \frac{\begin{array}{c} p \xrightarrow{E_p,k} p, k \neq 0 \\ \text{SE} \supset p \xrightarrow[E]{E_p,k} \text{SE} \cdot \supset p, \end{array}}{\text{SE} \supset p \xrightarrow[E]{E_p,k} \text{SE} \cdot \supset p,}
 \end{array}$$

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:*

$$1. \text{ nothing}; p \Leftrightarrow p.$$

$$2. p; \text{nothing} \Leftrightarrow p.$$

BEWEIS

Zu 1:

Es gilt:

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

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:

3.4. Die Transformation der Esterel-Befehle

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

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

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

$$\frac{i \in \{1, \dots, n\} \ SE_i \in E \wedge \forall j < i : SE_j \notin E \quad p_i \xrightarrow[E]{E_{p_i},k} p'_i}{SE_1, \dots, SE_n?p_1, \dots, p_n \xrightarrow[E]{E_{p_i},k} p'_i}$$

und

$$\frac{i \in \{1, \dots, n\} \ \forall i \leq n : SE_i \notin E \quad q \xrightarrow[E]{E_q,k} q'}{SE_1, \dots, SE_n?p_1, \dots, p_n \xrightarrow[E]{E_q,k} q'}$$

Lemma 2 (presentCase)

Es gilt :

$$SE_1, \dots, SE_n?p_1, \dots, p_n \iff SE_1?p_1, SE_2?p_2, \dots, SE_n?p_n, q$$

BEWEIS

Sei $i \in \{1, \dots, n\}$. 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 $SE_i \notin E$ gilt:

$$\frac{(\text{presentcase}) \quad i \in \{1, \dots, n\} \ \forall i \leq n : SE_i \notin E \quad q \xrightarrow[E]{E_q,k} q'}{SE_1, \dots, SE_n?p_1, \dots, p_n \xrightarrow[E]{E_q,k} q'}$$

ist äquivalent zu

$$\frac{\begin{array}{c} (\text{present}) SE_n \notin E \quad q \xrightarrow[E]{E_q,k} q' \\ \hline (\text{present}) SE_{n-1} \notin E \quad SE_n?p_n, q \xrightarrow[E]{E_q,k} q' \\ \vdots \quad \vdots \\ \hline (\text{present}) SE_1 \notin E \quad SE_2?p_2, \dots, SE_n?p_n, q \xrightarrow[E]{E_q,k} q' \end{array}}{SE_1?p_1, SE_2?p_2, \dots, SE_n?p_n, q \xrightarrow[E]{E_q,k} q'}$$

Für den Fall $SE_i \in E$ und $SE_j \notin E$ für alle $j < i$ gilt:

3. Transformation von Esterel in SyncCharts

$$\begin{array}{c}
 (\text{presentcase}) \quad i \in \{1, \dots, n\} \ SE_i \in E \wedge \forall j < i : SE_j \notin E \ p_i \xrightarrow[E]{E_{p_i,k}} p'_i \\
 \hline
 SE_1, \dots, SE_n ? p_1, \dots, p_n \xrightarrow[E]{E_{p_i,k}} p'_i
 \end{array}$$

ist äquivalent zu

$$\begin{array}{c}
 (\text{present}) SE_i \in E \qquad \qquad \qquad p_i \xrightarrow[E]{E_{p_i,k}} p'_i \\
 \hline
 (\text{present}) SE_{i-1} \in E \qquad \qquad \qquad SE_i ? p_i, q \xrightarrow[E]{E_{p_i,k}} p'_i \\
 \vdots \qquad \qquad \qquad \vdots \\
 \hline
 (\text{present}) SE_j \notin E \quad SE_{j+1} ? p_{j+1}, \dots, SE_n ? p_n, q \xrightarrow[E]{E_{p_i,k}} p'_i \\
 \vdots \qquad \qquad \qquad \vdots \\
 \hline
 SE_1 ? p_1, SE_2 ? p_2, \dots, SE_n ? p_n, q \xrightarrow[E]{E_{p_i,k}} p'_i
 \end{array}$$

Definition 12 (awaitcase) Die Verhaltensregel zu $DE_1, \dots, DE_n \not\Rightarrow p_1, \dots, p_n$ in der bereits etablierten Schreibweise lautet:

$$DE_1, \dots, DE_n \not\Rightarrow p_1, \dots, p_n \equiv (\{DE_1 \rightsquigarrow !S_1; 2 | \dots | DE_n \rightsquigarrow !S_n; 2\}; S_1, \dots, S_n ? p_1, \dots, p_n, 0) \\
 \setminus S_1 \dots \setminus S_n$$

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

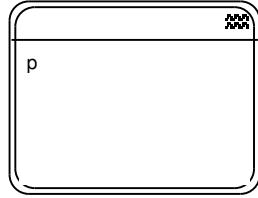
Sollte bei einem Esterel-Befehl in der beschreibenden Grammatik ein optionales $\langle Statement \rangle$ vorhanden sein, und an einer Stelle im Code von diesen optionalen $\langle Statement \rangle$ kein Gebrauch gemacht werden, so wird ein *nothing* als Ersatz verwendet. Durch diese Vereinfachung 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 folgenden Grammatik nicht verwendet. G sei die Transformationsgrammatik, V die Menge der Nichtterminal-Zeichen, **module** das Startsymbol und P die Menge der Transformationsregeln.

3.4. Die Transformation der Esterel-Befehle

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

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



$\Sigma =$ 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

$\langle \text{Module} \rangle ::= \text{'module'} \langle \text{Identifier} \rangle \text{':'} \{ \langle \text{DeclarationList} \rangle \} \langle \text{Statement} \rangle \text{'end'} [\text{'module'}]$

Beschreibung des Befehls

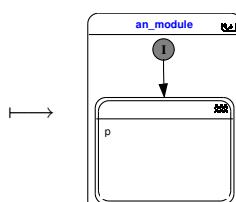
Ein **module** beschreibt den Namen eines Programms, die Namen externer Ein- und Ausgaben und die Syntax möglicher extern zu definierender Typen, Funktionen, Tasks und Prozeduren. Zudem werden Relationen zwischen den Signalen definiert. Die verschiedenen Deklarationen sind im Abschnitt 2.2.1 erläutert.

Beschreibung eines äquivalenten Makrozustandes

Die Deklarationen werden vom SyncChart übernommen. Zudem wird der Makrozustand, welcher aus der Transformation des $\langle \text{Statement} \rangle$ entsteht, von der Initialen Transition referenziert.

Transformationsregel 1 (module)

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



3. Transformation von Esterel in SyncCharts

Lemma 3 Es gilt die folgende Äquivalenz.

$\begin{array}{l} \text{module } <\text{Identifier}>: \\ <\text{DeclarationList}> \\ p \\ \text{end module} \end{array}$	\iff	$\begin{array}{l} \text{module } <\text{Identifier}>: \\ <\text{DeclarationList}> \\ \quad \text{nothing}; \\ \quad [p]; \\ \quad \text{halt} \\ \quad \text{end module} \end{array}$
--------------------------------------------------------------------------------------------------------------------------	--------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

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

$\langle Nothing \rangle ::= 'nothing'$

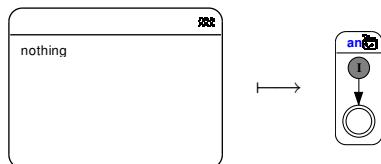
Beschreibung des Befehls

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 \iff *nothing*

BEWEIS In diesem Fall offensichtlich.

3.4.4. Der Befehl halt

Esterelgrammatik

$\langle Halt \rangle$::= ‘halt’

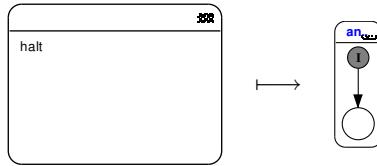
Beschreibung des Befehls

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.

$$\begin{array}{ccc} \textit{halt} & \iff & \textit{nothing;} \\ & & \textit{halt} \end{array}$$

BEWEIS Folgt aus Lemma 1

3.4.5. Der Befehl `pause`

Esterelgrammatik

$\langle \text{Pause} \rangle ::= \text{'pause'}$

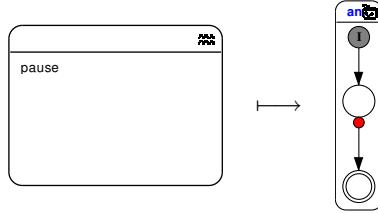
Beschreibung des Befehls

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

\iff

```

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 Abbruchbedingungen, so entspricht die textliche Reihenfolge der Priorität, wobei der textlich erste Fall, jener mit der höchsten Priorität ist.

Beschreibung eines äquivalenten Makrozustandes

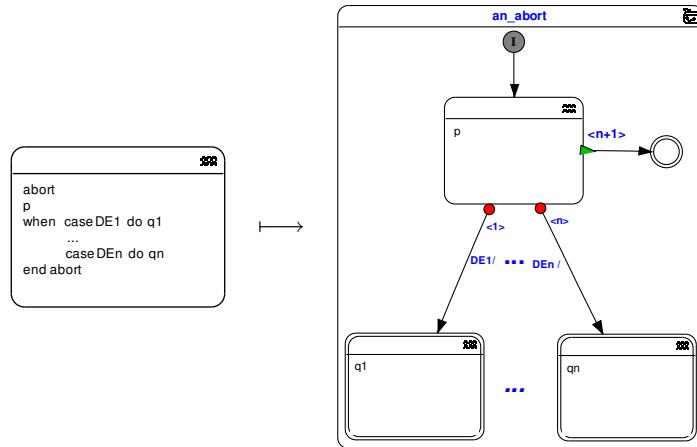
Um einen Makrozustand entsprechend der Semantik eines **abort**-Befehls abzubrechen, verwendet man *strong abortions* mit den entsprechenden *Delayexpressions* als

3.4. Die Transformation der Esterel-Befehle

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

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)

$$\text{weak abort } p \text{ when } DE \quad \mapsto \quad \text{weak abort } p \text{ when case } DE \text{ do nothing}$$

Lemma 7 Es gilt die folgende Äquivalenz.

$$\begin{array}{ccc}
 \begin{array}{l}
 \text{abort } p \\
 \text{case } DE1 \text{ do run } q1 \\
 \dots \\
 \text{case } DEn \text{ do run } qn \\
 \text{end abort}
 \end{array} & \iff &
 \begin{array}{l}
 \text{nothing;} \\
 \text{abort } [p]; \\
 \text{nothing} \\
 \text{when} \\
 \text{case } DE1 \text{ do} \\
 \text{nothing;} \\
 [q1] \\
 \dots \\
 \text{case } DEn \text{ do} \\
 \text{nothing;} \\
 [qn] \\
 \text{end abort}
 \end{array}
 \end{array}$$

BEWEIS Mit Hilfe des Lemma 1 ergibt sich die Äquivalenz.

3. Transformation von Esterel in SyncCharts

3.4.7. Der Befehl assign

Esterelgrammatik

$\langle \text{Assign} \rangle ::= \langle \text{VariableIdentifier} \rangle \text{ ':=' } \langle \text{Expression} \rangle$

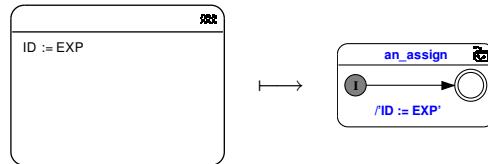
Beschreibung des Befehls

Dieser Befehl weist einer lokalen Variable eine $\langle \text{Expression} \rangle$ zu. Dazu müssen die Typen der $\langle \text{Expression} \rangle$ 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.

$$ID := EXP \iff \text{nothing}; ID := EXP$$

BEWEIS Hier folgt die Äquivalenz ebenfalls durch das Lemma 1.

3.4.8. Der Befehl await

Esterelgrammatik

$\langle \text{Await} \rangle ::= \text{'await'} \langle \text{DelayExpression} \rangle [\text{'do'} \langle \text{Statement} \rangle \text{'end'} [\text{'await'}]]$
 $\quad | \quad \text{'await case'} \langle \text{DelayExpression} \rangle [\text{'do'} \langle \text{Statement} \rangle]$
 $\quad \quad \{\text{'case'} \langle \text{DelayExpression} \rangle [\text{'do'} \langle \text{Statement} \rangle]\}$
 $\quad \text{'end'} [\text{'await'}]$

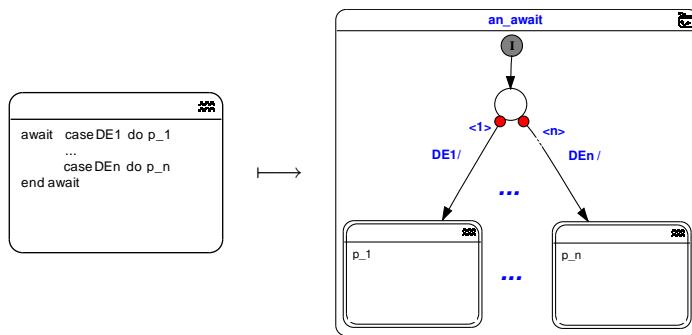
Beschreibung des Befehls

Der **await**-Befehl wartet auf das Eintreten einer $\langle \text{DelayExpression} \rangle$ und führt dann, wenn vorhanden, ein für diesen Fall vorgesehenes $\langle \text{Statement} \rangle$ aus. Stehen mehrere alternative *DelayExpressions* zur Auswahl, so wird zunächst die im Programmtext oberste überprüft und ggf. das $\langle \text{Statement} \rangle$ 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 Transition benötigt. 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)

$$\text{await } DE \text{ do } p \quad \mapsto \quad \text{await case } DE \text{ do } p$$

Lemma 9 Es gilt die folgende Äquivalenz.

$$\begin{array}{l}
 \text{nothing;} \\
 \text{await} \\
 \text{case } DE1 \text{ do } p_1 \\
 \dots \\
 \text{case } DEn \text{ do } p_n \\
 \text{end await}
 \end{array}
 \qquad \qquad \qquad
 \begin{array}{l}
 \text{nothing;} \\
 \text{await} \\
 \text{case } DE1 \text{ do} \\
 \text{nothing;} \\
 \iff [p_1] \\
 \text{case } DEn \text{ do} \\
 \text{nothing;} \\
 [p_n] \\
 \text{end await}
 \end{array}$$

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. Transformation von Esterel in SyncCharts

3.4.9. Der Befehl `doupto`

Esterelgrammatik

$\langle DoUpto \rangle ::= \text{'do'} \langle \text{Statement} \rangle \text{'upto'} \langle \text{SignalExpression} \rangle$

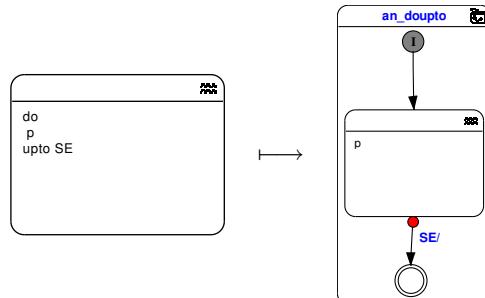
Beschreibung des Befehls

Es wird das $\langle \text{Statement} \rangle$ ausgeführt. Sollte dieses terminieren bevor die $\langle \text{SignalExpression} \rangle$ zutrifft, wird die weitere Ausführung angehalten und erst nach dem Auftreten der $\langle \text{SignalExpression} \rangle$ fortgesetzt. Tritt die $\langle \text{SignalExpression} \rangle$ während der Ausführung von $\langle \text{Statement} \rangle$ auf, so wird dieses sofort abgebrochen.

Beschreibung eines äquivalenten Makrozustandes

Das auszuführende $\langle \text{Statement} \rangle$ wird in einem Makrozustand dargestellt. Dieser Makrozustand kann nur über eine *strong abortion* mit der $\langle \text{SignalExpression} \rangle$ als Auslöser verlassen werden. Solange die $\langle \text{SignalExpression} \rangle$ 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.

$$\begin{array}{ccc} \text{do } p \\ \text{upto } SE & \iff & \begin{array}{c} \text{abort} \\ p; \text{halt} \\ \text{when } SE \end{array} \end{array}$$

Abbildung 3.1.: Die Alternative zu `doupto`

Lemma 10 Es gilt die folgende Äquivalenz.

$$\begin{array}{ccc}
 \textit{do } p & \xrightarrow{\quad} & \textit{nothing;} \\
 \textit{upto } S & & \textit{abort} \\
 & & \textit{[p];} \\
 & & \textit{halt} \\
 & & \textit{when } SE \\
 & & \textit{do} \\
 & & \textit{nothing} \\
 & & \textit{end abort}
 \end{array}$$

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

3.4.10. Der Befehl dowatching

Esterelgrammatik

$\langle \text{Do Watching} \rangle ::= \text{'do'} \langle \text{Statement} \rangle \text{'watching'} \langle \text{SignalExpression} \rangle [\text{'timeout'} \langle \text{Statement} \rangle \text{'end timeout'}]$

Beschreibung des Befehls

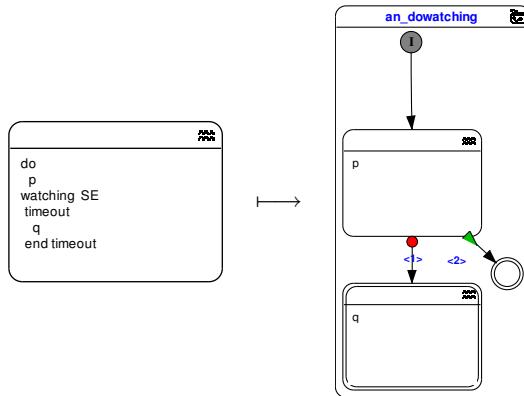
Beim Auftreten der $\langle \text{SignalExpression} \rangle$ wird das $\langle \text{Statement} \rangle$ sofort abgebrochen, ansonsten wird es normal ausgeführt. Terminiert das $\langle \text{Statement} \rangle$, so terminiert auch der **dowatching** Befehl. Existiert ein timeout-Konstrukt, so wird dieses unmittelbar nach dem Auftreten der $\langle \text{SignalExpression} \rangle$ gestartet.

Beschreibung eines äquivalenten Makrozustandes

Das $\langle \text{Statement} \rangle$ 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 $\langle \text{SignalExpression} \rangle$ eintritt. Die *strong abortion* endet an einem finalen Makrozustand, welches in einem timeout-Fall das entsprechende, auszuführende $\langle \text{Statement} \rangle$ oder ein **nothing** darstellt.

3. Transformation von Esterel in SyncCharts

Transformationsregel 11 (dowatching)



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

<pre> do p watching SE timeout q end timeout </pre>	<pre> abort p when SE do q end abort </pre>
-----------------------------------------------------	---------------------------------------------

Abbildung 3.2.: Die Alternative zu `dowatching`

Lemma 11 Es gilt die folgende Äquivalenz.

<pre> do p watching SE timeout dotimeout end timeout </pre>	\iff	<pre> nothing; abort [p]; nothing when [tick] do nothing; [q] end abort } </pre>
-------------------------------------------------------------	--------	----------------------------------------------------------------------------------

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

3.4.11. Der Befehl emit

Esterelgrammatik

$\langle \text{Emit} \rangle ::= \text{'emit'} \langle \text{SignalIdentifier} \rangle [('Expression')]$

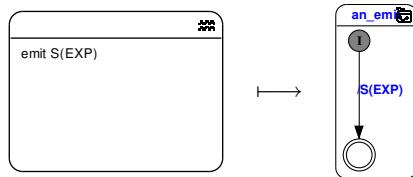
Beschreibung des Befehls

emit sendet ein Signal, wobei je nach Typ des Signals der Wert einer $\langle \text{Expression} \rangle$ 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.

$$\text{emit } S(\text{EXP}) \iff \text{emit } S(\text{EXP})$$

BEWEIS Die linke und die rechte Seite sind offensichtlich äquivalent.

3.4.12. Der Befehl every

Esterelgrammatik

$\langle \text{Every} \rangle ::= \text{'every'} \langle \text{DelayExpresion} \rangle \text{'do'} \langle \text{Statement} \rangle \text{'end'} [\text{'every'}]$

Beschreibung des Befehls

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

3. Transformation von Esterel in SyncCharts

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)

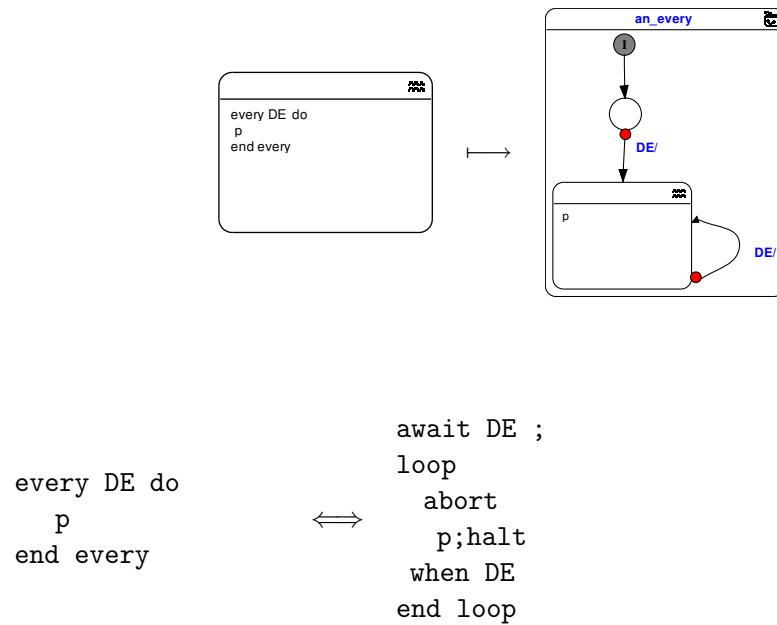


Abbildung 3.3.: Alternative zu **every**

Lemma 13 Es gilt die folgende Äquivalenz.

<code>nothing;</code> <code>await case [DE] do</code> <code>nothing</code> <code>end await ;</code> <code>loop</code> <code> abort</code> <code> [p];</code> <code> halt</code> <code> when [DE]</code> <code> do</code> <code> nothing</code> <code> end abort</code> <code> end loop</code>	\iff	<code>every DE do</code> <code> p</code> <code>end every</code>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------	--------------------------------------------------------------------------

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

3.4.13. Der Befehl if

Esterelgrammatik

$\langle If \rangle$	$::= \text{'if'} \langle \text{Expression} \rangle [\text{'then'} \langle \text{Statement} \rangle] [\langle \text{ElseIf} \rangle \{ \langle \text{ElseIf} \rangle \}] [\langle \text{else} \rangle \langle \text{Statement} \rangle] \text{'end'} \text{'if'}$
$\langle \text{ElseIf} \rangle$	$::= \text{'elsif'} \langle \text{Expression} \rangle [\text{'then'} \langle \text{Statement} \rangle]$

Beschreibung des Befehls

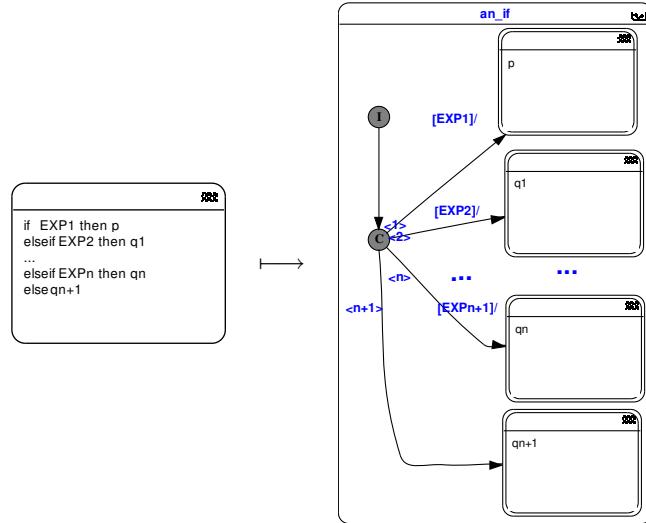
Der **if**-Befehl reagiert auf die Auswertung der `<Expression>`s. Sollte eine `<Expression>` zutreffen, so wird das zugehörige `<Statement>` sofort ausgeführt und keine weitere Auswertung der folgenden `<Expression>`s vorgenommen. Die Reihenfolge der Auswertung entspricht der textlichen Reihenfolge. 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.

3. Transformation von Esterel in SyncCharts

Transformationsregel 14 (if)



Lemma 14 Es gilt die folgende Äquivalenz.

$$\begin{array}{c}
\text{nothing;} \\
\text{if } EXP1 \text{ then} \\
\quad \text{nothing;} \\
\quad [p] \\
\text{elseif } EXP2 \text{ then} \\
\quad \text{nothing;} \\
\quad [q_1] \\
\cdots \\
\text{elseif } EXP_n \text{ then } q_n \\
\text{else } q_{n+1} \\
\text{end if} \\
\iff \\
\text{nothing;} \\
\text{elseif } EXP_n \text{ then} \\
\quad \text{nothing;} \\
\quad [q_n] \\
\text{else} \\
\quad \text{nothing;} \\
\quad [q_{n+1}] \\
\text{end if}
\end{array}$$

BEWEIS Die Behauptung gilt aufgrund des Lemma 1.

3.4.14. Der Befehl localsignal

Esterelgrammatik

$\langle LocalSignal \rangle ::= \text{'signal'} \langle SignalDeclList \rangle \text{'in'} \langle Statement \rangle [\text{'end signal'}]$

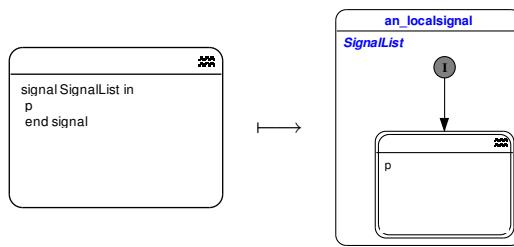
Beschreibung des Befehls

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>` entsprechend ein, so kann man diese Signale im Makrozustand und allen, in der Hierarchie folgenden Zuständen nutzen.

Transformationsregel 15 (localsignal)



Lemma 15 Es gilt die folgende Äquivalenz.

$$\begin{array}{ccc}
 \text{signal } \text{SignalList} \text{ in} & & \text{signal } \text{SignalList} \\
 \text{p} & \Longleftrightarrow & \text{in} \\
 \text{end signal} & & \text{nothing;} \\
 & & [p] \\
 & & \text{end signal}
 \end{array}$$

BEWEIS Die Behauptung folgt aus dem Lemma 1.

3.4.15. Der Befehl localvariable

Esterelgrammatik

$\langle \text{LocalVariable} \rangle ::= \text{'var'} \langle \text{VariableDeclarationList} \rangle \text{'in'}$
 $\quad \langle \text{Statement} \rangle$
 $\quad \text{'end'} \text{ '['var'] }$

$\langle \text{VariableDeclarationList} \rangle ::= \langle \text{Identifier} \rangle [':=\langle \text{Expression} \rangle] ':' \langle \text{TypeIdentifier} \rangle$
 $\quad \{ ; \langle \text{Identifier} \rangle [':=\langle \text{Expression} \rangle] ':' \langle \text{TypeIdentifier} \rangle \}$

3. Transformation von Esterel in SyncCharts

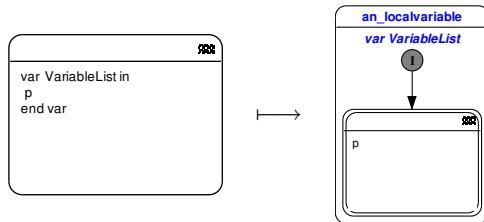
Beschreibung des Befehls

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.

$$\begin{array}{ccc}
 \text{var } \textit{VariableList} \text{ in} & & \text{var } \textit{VariableList} \\
 \textit{p} & \iff & \text{in} \\
 \textit{end var} & & \textit{nothing}; \\
 & & [\textit{p}] \\
 & & \textit{end var}
 \end{array}$$

BEWEIS Die Behauptung folgt aus dem Lemma 1.

3.4.16. Der Befehl loop

Esterelgrammatik

$\langle \text{Loop} \rangle ::= \text{'loop'} \text{ } \langle \text{Statement} \rangle \text{ } [\text{'end loop'}]$

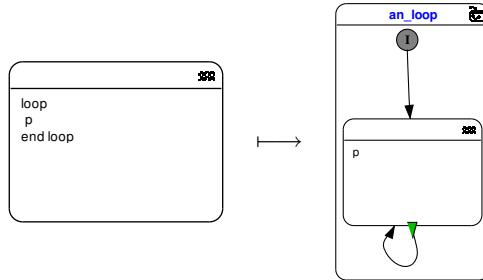
Beschreibung des Befehls

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.

$$\begin{array}{ccc}
 \text{loop} & & \text{nothing;} \\
 \text{p} & \iff & \text{loop} \\
 \text{end loop} & & [\text{p} \\
 & &] ; \\
 & & \text{nothing} \\
 & & \text{end loop}
 \end{array}$$

BEWEIS Die Behauptung gilt aufgrund von Lemma 1.

3.4.17. Der Befehl loopeach

Esterelgrammatik

$\langle \text{LoopEach} \rangle ::= \text{'loop'} \langle \text{Statement} \rangle \text{'each'} \langle \text{DelayExpression} \rangle$

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. Terminierte das <Statement> vor dem Auslösen der <DelayExpression>, so wird auf das Eintreten der <DelayExpression> gewartet. Eine alternative 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 abortion*-

3. Transformation von Esterel in SyncCharts

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)

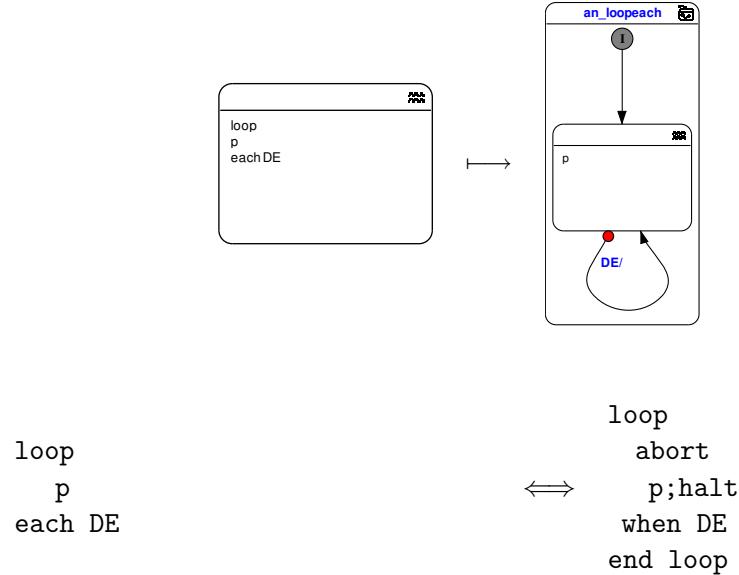


Abbildung 3.4.: Alternative zu loopeach

Lemma 18 Es gilt die folgende Äquivalenz.

$$\begin{array}{ccc}
 \text{loop} & & \text{loop} \\
 \text{p} & \iff & \text{abort} \\
 \text{each DE} & & \text{when DE} \\
 & & \text{end loop}
 \end{array}
 \quad
 \begin{array}{c}
 \text{nothing;} \\
 \text{loop} \\
 \text{abort} \\
 [p]; \\
 \text{halt} \\
 \text{do} \\
 \text{nothing} \\
 \text{end abort} \\
 \text{end loop}
 \end{array}$$

BEWEIS Es gilt die folgende Äquivalenz.

3.4.18. Der Befehl parallel

Esterelgrammatik

```
 $\langle Parallel \rangle ::= (\langle NonParallel \rangle$ 
 $\quad | \quad \langle Parallel \rangle) '||' \langle NonParallel \rangle$ 
```

```
 $\langle NonParallel \rangle ::= \langle AtomicStatement \rangle$ 
 $\quad | \quad \langle Sequence \rangle$ 
```

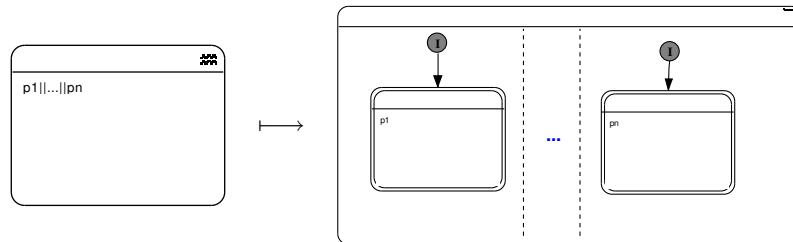
Beschreibung des Befehls

Durch das `||` Kommando können beliebig viele nicht-parallele Befehle $p_i \ i \in 0, \dots, n$ unmittelbar parallel ausgeführt werden. Das `||` Kommando 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.

$$\begin{array}{ccc}
 & \xrightarrow{\quad} & \\
 \begin{array}{c} [\text{nothing}; \\ \quad [p1] \\ \quad / / \\ p1 \ / / \dots \ / / \ pn \end{array} & \xrightleftharpoons{\quad} & \begin{array}{c} \dots \\ / / \\ \begin{array}{c} \text{nothing}; \\ \quad [pn] \\ \quad] \end{array} \end{array}
 \end{array}$$

BEWEIS Die Behauptung folgt unter Einbeziehung von Lemma 1.

3. Transformation von Esterel in SyncCharts

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.

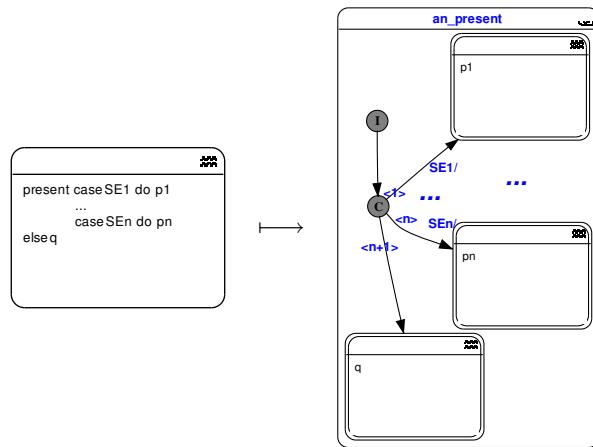
- `present SE then p else q end present`
- `present case SE1 do p1 ... case SEn do pn else q end present`

Die erste Variante ist nur eine abkürzende Schreibweise der zweiten für den Fall, dass nur eine Bedingung 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 entspricht 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 3.4.13, nur werden die `<SignalExpression>`s als *Trigger* nicht als Kondition verwendet.

Transformationsregel 20 (present)



Transformationsregel 21 (then in case)
 $\text{then} \quad \mapsto \quad \text{case}$

Lemma 20 Es gilt die folgende Äquivalenz.

$\text{present case } SE_1 \text{ do } p_1$ \dots $\quad \text{case } SEn \text{ do } pn$ $\quad \text{else } q$ end present	\iff	nothing; present $\text{case } SE_1 \text{ do}$ $\quad \text{nothing;}$ $\quad [p_1]$ \dots $\text{case } SEn \text{ do}$ $\quad \text{nothing;}$ $\quad [pn]$ else $\quad \text{nothing;}$ $\quad [q]$ end present
-----------------------------------------------------------------------------------------------------------------------------------------------------	--------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

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

3.4.20. Der Befehl call

Esterelgrammatik

$\langle Call \rangle ::= \text{'call'} \langle \text{ProcedureIdentifier} \rangle \text{('} \{ \langle \text{VariableIdentifier} \rangle \} \text{')} \text{('} \langle \text{Expression} \rangle \text{')}'$

3. Transformation von Esterel in SyncCharts

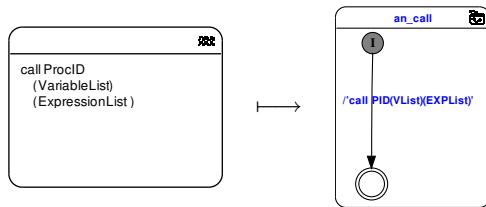
Beschreibung des Befehls

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.

$$\text{call } PID(VList)(EXPList) \iff \begin{array}{l} \text{nothing;} \\ \text{call } PID(VList)(EXPList) \end{array}$$

BEWEIS Die Behauptung folgt aus dem Lemma 1.

3.4.21. Der Befehl sequence

Esterelgrammatik

$\langle Sequence \rangle ::= \langle SequenceWithoutTerminator \rangle [';']$

$\langle SequenceWithoutTerminator \rangle ::= \langle AtomicStatement \rangle ; \langle AtomicStatement \rangle$
 $| \quad \langle SequenceWithoutTerminator \rangle ; \langle AtomicStatement \rangle$

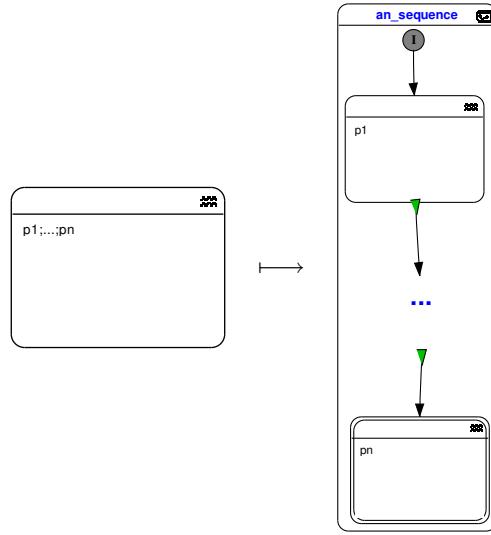
Beschreibung des Befehls

Eine Sequenz von $\langle AtomicStatement \rangle ; \dots ; \langle AtomicStatement \rangle$ wird sofort ausgeführt. Terminiert ein $\langle AtomicStatement \rangle$, so wird sofort das Folgende ausgeführt. Terminiert das letzte $\langle AtomicStatement \rangle$, so terminiert die Sequenz ebenfalls. Wird innerhalb der Sequenz eine, der Sequenz übergeordnete, Trap ausgelöst, so werden nachfolgende $\langle AtomicStatement \rangle$ 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.

$$\begin{array}{ccc}
 & \text{nothing;} & \\
 & [p_1]; & \\
 p_1; \dots; p_n & \iff & \dots \\
 & \text{nothing;} & \\
 & [p_n]; &
 \end{array}$$

BEWEIS Aufgrund von Lemma 1 ist die Behauptung wahr.

3.4.22. Der Befehl suspend

Esterelgrammatik

$\langle Suspend \rangle ::= \text{'suspend'} \langle \text{Statement} \rangle \text{'when'} \text{[}'\text{immediate'}\text{]} \langle \text{SignalExpression} \rangle$

Beschreibung des Befehls

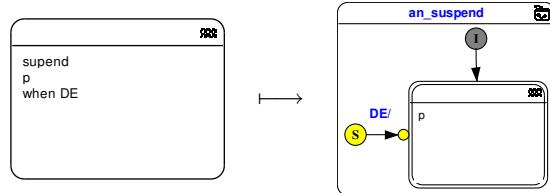
Der **suspend**-Befehl unterbricht die Ausführung des `<Statement>`s, solange die `<SignalExpression>` gültig ist.

3. Transformation von Esterel in SyncCharts

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.

$$\begin{array}{ccc}
 \text{suspend} & & \text{nothing}; \\
 p & \iff & \text{suspend} \\
 \text{when } DE & & [p] \\
 & & \text{when } DE
 \end{array}$$

BEWEIS Die Behauptung trifft aufgrund des Lemma 1 zu.

3.4.23. Der Befehl sustain

Esterelgrammatik

$\langle Sustain \rangle ::= \text{'sustain'} \langle \text{SignalIdentifier} \rangle [(' \langle \text{Expression} \rangle ')]$

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

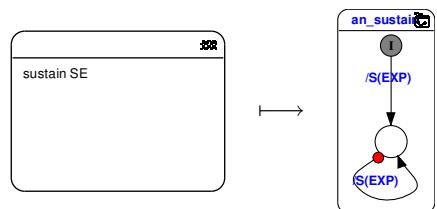
Ein einfacher Zustand wird durch eine *strong abortion* in jeder Instanz abgebrochen, wobei <SignalIdentifier> [(' <Expression> ') gesendet wird. Das Ziel der *strong abortion* ist wieder der einfache Zustand. Zusätzlich muß beim Übergang vom Initialzustand zum einfachen Zustand das Signal <SignalIdentifier>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

3.4. Die Transformation der Esterel-Befehle

sustain S(EXP) loop
 \iff emit S(EXP)
 each tick

Abbildung 3.5.: Alternative zu **sustain**

Transformationsregel 25 (sustain)



3. Transformation von Esterel in SyncCharts

Lemma 24 Es gilt die folgende Äquivalenz.

```

emit S(EXP);
loop
  await case [tick] do
    emit S(EXP)
  end await
end loop
↔
sustain S(EXP)

```

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 Ableitung, die ich mit Hilf1 abkürzen wird:

$$\begin{array}{c} \left(\left(\left\{ \text{tick} \Rightarrow !S_1; 2 \right\}; S_1 ? S(\text{EXP}), 0 \right) \setminus S_1 \right)^* \xrightarrow[E]{\emptyset, 1} \left(\left\{ 0; \text{tick} \Rightarrow (!S_1; 2) \right\}; S_1 ? S(\text{EXP}), 0 \right) \setminus S_1; \\ \left(\left(\left\{ \text{tick} \Rightarrow !S_1; 2 \right\}; S_1 ? S(\text{EXP}), 0 \right) \setminus S_1 \right)^* \end{array}$$

BEWEIS
Es gilt:

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

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

(seq2)) !S(EXP); (tick \Rightarrow !S(EXP))^{*} $\xrightarrow[E]{S(EXP),1} (\{0;\text{tick}\cdot \Rightarrow(!S_1;2)\}; S_1?S(EXP),0) \ S_1; ((({\text{tick}}\Rightarrow !S_1;2) ; S_1?S(EXP),0)) \ S_1)^*$

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

3. Transformation von Esterel in SyncCharts

$(\text{emit})!S_1 \xrightarrow[E]{S_{1,0}} 0$	$(\text{exit}) 2 \xrightarrow[E]{\emptyset,2} 0$
$(\text{seq2})!S_1;2 \xrightarrow[E]{S_{1,2}} 0$	
$(\text{shift})\uparrow(!S_1;2) \xrightarrow[E]{S_{1,3}} \uparrow(0)$	
$(\text{present+})(\text{tick})?(\uparrow(!S_1;2);1) \xrightarrow[E]{S_{1,3}} \uparrow(0)$	
$(\text{loop})((\text{tick})?(\uparrow(!S_1;2),1))^* \xrightarrow[E]{S_{1,3}} \uparrow(0);(\uparrow(!S_1;2),1))^*$	
$(\text{trap2})\{((\text{tick})?(\uparrow(!S_1;2),1))^* \} \xrightarrow[E]{S_{1,2}} \{\uparrow(0);(\uparrow(!S_1;2),1))^*\}$	$(\text{emit})!S(\text{EXP}) \xrightarrow[E]{S(\text{EXP}),0} 0$
$(\text{trap})(\text{nothing}) \{0;\text{tick}\cdot\Rightarrow(\uparrow(!S_1;2))\} \xrightarrow[E]{S_{1,0}} 0$	$(\text{present+})S_1?S(\text{EXP}),0 \xrightarrow[E]{S(\text{EXP}),0} 0$
$(\text{seq2})\{0;\text{tick}\cdot\Rightarrow(\uparrow(!S_1;2)); S_1?S(\text{EXP}),0\} \xrightarrow[E]{S_{1*}S(\text{EXP}),0} 0$	
$(\text{sig+})\{0;\text{tick}\cdot\Rightarrow(\uparrow(!S_1;2)); S_1?S(\text{EXP}),0\} \setminus S_1 \xrightarrow[E]{S(\text{EXP}),0} (0) \setminus S_1$	$(\text{Lemma 25}) \text{ Hilf1}$
$(\text{seq2})\{0;\text{tick}\cdot\Rightarrow(\uparrow(!S_1;2)); S_1?S(\text{EXP}),0\} \setminus S_1; \xrightarrow[E]{S(\text{EXP}),1} (\{0;\text{tick}\cdot\Rightarrow(\uparrow(!S_1;2)); S_1?S(\text{EXP}),0\} \setminus (((\{\text{tick}\Rightarrow\uparrow(!S_1;2)\}; S_1?S(\text{EXP}),0) \setminus S_1)^*)$	

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

3.4.24. Der Befehl trap

Esterelgrammatik

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

```
<ExceptionDeclaration> ::= <ExceptionIdentifier> [| ':=' <Expression>]
                           | ':' <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-Verknüpfungen `<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*,

3. Transformation von Esterel in SyncCharts

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

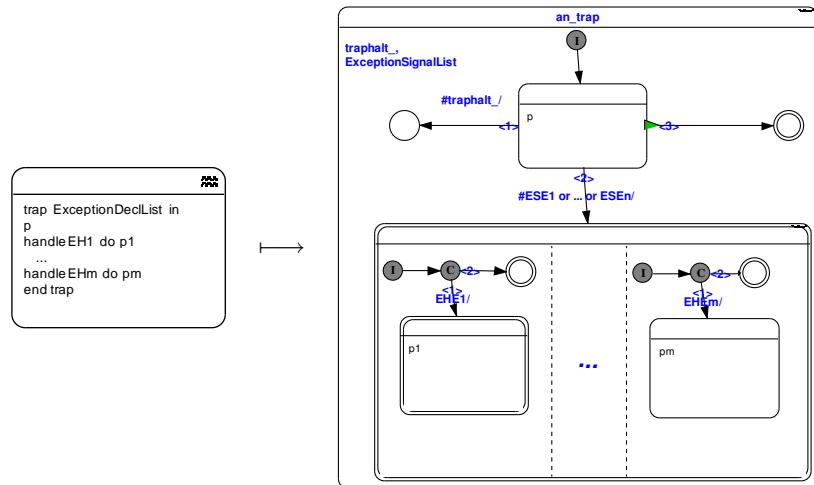
Der erste Punkt wird mit Hilfe des **traphalt**-Signals gelöst, indem dieses Signal beim Auslösen eines Ausnahmesignals einer priorisierten **trap** durch **exit** 3.4.25 zusätzlich gesendet wird. Daher werden neben dem Ausnahmesignal alle **traphalt**-Signale der Umwandlung eines **exit**-Befehls gesendet, 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
*	beliebig

Lemma 26 (H1)

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

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

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

BEWEIS Es gilt:

$$(1) \quad S \in E$$

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

$$(2) \quad S \notin E$$

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

Mit Lemma 1 folgt die Behauptung.

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

$$(1) \quad p \text{ terminiert nicht, also } k_{r_3} = 1 \text{ und } S_1, S_2 \notin E_p. \text{ Dann gilt}$$

$$\{\uparrow p;2 \mid \{r_1\} \mid \{r_2\}\} \xrightarrow[E]{E_p,1} \{\uparrow p';2 \mid \{r_1\} \mid \{r_2\}\}$$

$$(2) \quad p \text{ terminiert, } S_1, S_2 \in E_p. \text{ Dann gilt } \{\uparrow p;2 \mid \{r_1\} \mid \{r_2\}\} \xrightarrow[E]{E_p,0} 0$$

$$(3) \quad S_1, S_2 \in E_p, \text{ dann gilt } \{\uparrow p;2 \mid \{r_1\} \mid \{r_2\}\} \xrightarrow[E]{E_p * \{HS_1\} * \{HS_2\},0} 0$$

3. Transformation von Esterel in SyncCharts

$$(4) \quad S_1 \in E_p \text{ und } S_2 \notin E_p, \text{ dann gilt } \{\uparrow p; 2 | \{r_1\} \mid \{r_2\}\} \xrightarrow[E]{E_p * \{HS_1\}, 0} 0$$

$$(5) \quad S_1 \notin E_p \text{ und } S_2 \in E_p, \text{ dann gilt } \{\uparrow p; 2 | \{r_1\} \mid \{r_2\}\} \xrightarrow[E]{E_p * \{HS_2\}, 0} 0$$

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

Es gilt:

$$(1) \quad p \text{ terminiert nicht, also } k_{r_3} = 1 \text{ und } S_1, S_2 \notin E_p.$$

Es gilt:

$$\frac{\begin{array}{c} \uparrow p; 2 \xrightarrow[E]{E_p, 1} \uparrow p'; 2 \\ (\text{H1})\{r_1\} \xrightarrow[E]{\emptyset, 0} \{r_1\} \quad (\text{H1})\{r_2\} \xrightarrow[E]{\emptyset, 0} \{r_2\} \end{array}}{\begin{array}{c} (\text{parallel}) \uparrow p; 2 | \{r_1\} \mid \{r_2\} \xrightarrow[E]{E_p, 1} \uparrow p'; 2 | \{r_1\} \mid \{r_2\} \\ (\text{trap2}) \{\uparrow p; 2 | \{r_1\} \mid \{r_2\}\} \xrightarrow[E]{E_p, 1} \{\uparrow p'; 2 | \{r_1\} \mid \{r_2\}\} \end{array}}$$

$$(2) \quad p \text{ terminiert, } S_1, S_2 \in E_p.$$

Dann gilt:

$$\frac{\begin{array}{c} \uparrow p; 2 \xrightarrow[E]{E_p, 2} 0 \\ (\text{H1})\{r_1\} \xrightarrow[E]{\emptyset, 0} \{r_1\} \quad (\text{H1})\{r_2\} \xrightarrow[E]{\emptyset, 0} \{r_2\} \end{array}}{\begin{array}{c} (\text{parallel}) \uparrow p; 2 | \{r_1\} \mid \{r_2\} \xrightarrow[E]{E_p, 2} 0 | \{r_1\} \mid \{r_2\} \\ (\text{trap1}) \{\uparrow p; 2 | \{r_1\} \mid \{r_2\}\} \xrightarrow[E]{E_p, 0} 0 \end{array}}$$

$$(3) \quad S_1, S_2 \in E_p.$$

Dann gilt:

$$\frac{\begin{array}{c} \uparrow p; 2 \xrightarrow[E]{E_p, k_{r_3} < 3} \star \quad (\text{H1})\{r_1\} \xrightarrow[E]{\{HS_1\}, 2} \{\uparrow 0; 2; r_1\} \quad (\text{H1})\{r_2\} \xrightarrow[E]{\{HS_2\}, 2} \{\uparrow 0; 2; r_2\} \\ (\text{parallel}) \uparrow p; 2 | \{r_1\} \mid \{r_2\} \xrightarrow[E]{E_p * \{HS_1\} * \{HS_2\}, 2} \star | \{\uparrow 0; 2; r_1\} \mid \{\uparrow 0; 2; r_2\} \end{array}}{\begin{array}{c} (\text{trap1}) \{\uparrow p; 2 | \{r_1\} \mid \{r_2\}\} \xrightarrow[E]{E_p * \{HS_1\} * \{HS_2\}, 0} 0 \end{array}}$$

$$(4) \quad S_1 \in E_p \text{ und } S_2 \notin E_p.$$

Dann gilt:

$$\frac{\begin{array}{c} \uparrow p; 2 \xrightarrow[E]{E_p, k_{r_3} < 3} \star \quad (\text{H1})\{r_1\} \xrightarrow[E]{\{HS_1\}, 2} \{\uparrow 0; 2; r_1\} \quad (\text{H1})\{r_2\} \xrightarrow[E]{\emptyset, 0} \{r_2\} \\ (\text{parallel}) \uparrow p; 2 | \{r_1\} \mid \{r_2\} \xrightarrow[E]{E_p * \{HS_1\}, 2} \star | \{\uparrow 0; 2; r_1\} \mid \{r_2\} \end{array}}{\begin{array}{c} (\text{trap1}) \{\uparrow p; 2 | \{r_1\} \mid \{r_2\}\} \xrightarrow[E]{E_p * \{HS_1\}, 0} 0 \end{array}}$$

3.4. Die Transformation der Esterel-Befehle

(5) $S_1 \notin E_p$ und $S_2 \in E_p$.

Dann gilt

$$\frac{\begin{array}{c} \uparrow p; 2 \xrightarrow[E]{E_p, k_{r_3} < 3} \star \\ (H1)\{r_1\} \xrightarrow[E]{\emptyset, 0} \{r_1\} \quad (H1)\{r_2\} \xrightarrow[E]{\{HS_2\}, 2} \{\uparrow 0; 2; r_2\} \end{array}}{\begin{array}{c} (\text{parallel}) \uparrow p; 2 | \{r_1\} | \{r_2\} \xrightarrow[E]{E_p * \{HS_2\}, 2} \star | \{r_1\} | \{\uparrow 0; 2; r_2\} \\ (\text{trap1}) \{\uparrow p; 2 | \{r_1\} | \{r_2\}\} \xrightarrow[E]{E_p * \{HS_2\}, 0} 0 \end{array}}$$

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 $S_1 \in E$. Es gilt:

$$\frac{\begin{array}{c} (\text{emit}) !HS_2 \xrightarrow[E]{\{HS_2\}, 0} 0; \quad (\text{emit}) !HS_1 \xrightarrow[E]{\{HS_1\}, 0} 0 \\ \hline (\text{seq}) !HS_2; !HS_1 \xrightarrow[E]{\{HS_2, HS_1\}, 0} 0 \end{array}}{(\text{presentcase}) S_1, S_2 ? (!HS_2; !HS_1), (!HS_2; !HS_3) , 0 \xrightarrow[E]{\{HS_2, HS_1\}, 0} 0}$$

(2) Ist $S_2 \in E$ $S_1 \notin E$, dann gilt Es gilt:

$$\frac{\begin{array}{c} (\text{emit}) !HS_2 \xrightarrow[E]{\{HS_2\}, 0} 0; \quad (\text{emit}) !HS_3 \xrightarrow[E]{\{HS_3\}, 0} 0 \\ \hline (\text{seq}) !HS_2; !HS_3 \xrightarrow[E]{\{HS_2, HS_3\}, 0} 0 \end{array}}{(\text{presentcase}) S_1, S_2 ? (!HS_2; !HS_1), (!HS_2; !HS_3) , 0 \xrightarrow[E]{\{HS_2, HS_3\}, 0} 0}$$

(3) Ist $S_1, S_2 \notin E$, dann gilt Es gilt:

$$\frac{\begin{array}{c} (\text{null}) 0 \xrightarrow[E]{\emptyset, 0} 0 \\ \hline (\text{presentcase}) S_1, S_2 ? (!HS_2; !HS_1), (!HS_2; !HS_3) , 0 \xrightarrow[E]{\emptyset, 0} 0 \end{array}}$$

Lemma 29 (H4) Seien $S_1, S_2, HS_1, HS_2, HS_3$ Signale und sei

$r := (\{\uparrow p; 2 | \text{immediate } S_1, \text{immediate } S_2 \Rightarrow (!HS_1; !HS_2) \text{ } (!HS_1; !HS_3)\}) \setminus HS_1$, dann gilt:

3. Transformation von Esterel in SyncCharts

- (1) p terminiert nicht und $S_1, S_2 \notin E_p$. $r \xrightarrow[E]{E_p, 0} (\{ \uparrow p \}; 2 \mid immediate S_1, immediate S_2 \Rightarrow (!HS_1; !HS_2) (!HS_1; !HS_3)) \setminus HS_1$
- (2) p terminiert, $S_1, S_2 \in E_p$. Dann gilt $r \xrightarrow[E]{E_p * \{HS_2\}, 0} 0 \setminus HS_1$
- (3) $S_1, S_2 \in E_p$, dann gilt $r \xrightarrow[E]{E_p * \{HS_2\}, 0} 0 \setminus HS_1$
- (4) $S_1 \in E_p$ und $S_2 \notin E_p$, dann gilt $r \xrightarrow[E]{E_p * \{HS_2\}, 0} 0 \setminus HS_1$
- (5) $S_1 \notin E_p$ und $S_2 \in E_p$, dann gilt $r \xrightarrow[E]{E_p * \{HS_3\}, 0} 0 \setminus HS_1$

BEWEIS (1) p terminiert nicht und $S_1, S_2 \notin E_p$. Die Behauptung folgt aus dem Lemma 27.

- (2) p terminiert, $S_1, S_2 \in E_p$. Die Behauptung folgt aus dem Lemma 27 und Lemma 28 und der Verhaltensregel (signal).
- (3) $S_1, S_2 \in E_p$. Die Behauptung folgt aus dem Lemma 27 und Lemma 28 und der Verhaltensregel (signal).
- (4) $S_1 \in E_p$ und $S_2 \notin E_p$. Die Behauptung folgt aus dem Lemma 27 und Lemma 28 und der Verhaltensregel (signal).
- (5) $S_1 \notin E_p$ und $S_2 \in E_p$. Die Behauptung folgt aus dem Lemma 27 und Lemma 28 und der Verhaltensregel (signal).

Lemma 30 (H5) Sei S ein Signal.

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

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

BEWEIS

- (1) Ist $S \in E$, so folgt
$$\frac{\begin{array}{c} (\text{unit delay}) 1 \xrightarrow[E]{\emptyset, 1} 0 \\ \hline (\text{loop}) 1^* \xrightarrow[E]{\emptyset, 1} 0; 1^* \end{array}}{(\text{present+}) S ? 1^*, 0 \xrightarrow[E]{\emptyset, 1} 0; 1^*}$$

Die Behauptung folgt mit Lemma 1.
- (2) Ist $S \notin E$, so folgt
$$\frac{(\text{null}) 0 \xrightarrow[E]{\emptyset, 0} 0}{(\text{present-}) S ? 1^*, 0 \xrightarrow[E]{\emptyset, 0} 0}$$

3.4. Die Transformation der Esterel-Befehle

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

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

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

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

Lemma 32 Es gilt die folgende Äquivalenz.

3. Transformation von Esterel in SyncCharts

```

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

```

↔

```

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

```

3.4. Die Transformation der Esterel-Befehle

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

$$(H_4)(\{\uparrow p; 2 \text{ immediate th, immediate } ES \Rightarrow (w1;0;go20), (w1;0;go23)\}) \backslash w1 \xrightarrow[E]{E_p * go20,0} 0 \quad (H_5)(go20?1^*, 0) \xrightarrow[E]{E_p * go20,0} 1^*$$

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

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

$$(H_4)(\{\uparrow p; 2 \text{ immediate th, immediate } ES \Rightarrow (w1;0;go20), (w1;0;go23)\}) \backslash w1 \xrightarrow[E]{E_p * go23,0} 0$$

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

Aufgrund der Sequenz folgt nun die Ausführung aller ausgelösten *Handler* p , wobei für $EHEi?0;p_i;0 \xrightarrow[E]{\emptyset,0} 0$ gilt falls $EHEi \notin E$

$$\text{und } EHEi?0;p_i;0 \xrightarrow[E]{E_{pi},k} p_i \text{ falls } EHEi \in E$$

$$(0; EHE1?0;p_1;0), 0 \mid \dots \mid ((0; EHEm?0;p_m;0), 0) \xrightarrow[E]{E_{p1} * E_{pm}, k} p_1 \mid \dots \mid p_m.$$

$$\text{go23?}((0; EHE1?0;p_1;0), 0) \mid \dots \mid ((0; EHEm?0;p_m;0), 0) \xrightarrow[E]{E_{p1} * E_{pm}, k} p_1 \mid \dots \mid p_m$$

Sind alle p_i terminiert so folgt $(0 \mid \dots \mid 0) \xrightarrow[E]{E_{p1} * E_{pm}, 0} 0$ 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 entfernen der lokalen Signale.

3. Transformation von Esterel in SyncCharts

3.4.25. Der Befehl exit

Esterelgrammatik

$\langle \text{Exit} \rangle ::= \text{'exit'} \langle \text{ExceptionIdentifier} \rangle [('' \langle \text{Expression} \rangle '')]$

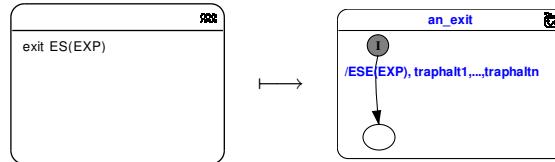
Beschreibung des Befehls

Der **exit**-Befehl emittiert ein Ausnahmesignal, welches eine **trap** auslöst. Das Ausnahmesignal verhält sich wie ein normales Signal. Eventuell vorhandene Sequenzen, die in der Hierarchie 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 Hierarchie 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.

$$\begin{array}{ccc}
 \text{exit ESE(EXP)} & \iff & \begin{array}{l}
 \text{emit ESE;} \\
 \text{emit traphalt1;} \\
 \dots \\
 \text{emit traphaltn;} \\
 \text{halt}
 \end{array}
 \end{array}$$

BEWEIS Es gilt aufgrund von mehreren Anwendungen von (**emit**).

$$!\text{ESE}; !\text{traphalt}_1; \dots; !\text{traphalt}_n; 1^* \xrightarrow[E]{\{ESE\} * \{\text{traphalt}_1\} * \dots * \{\text{traphalt}_n\}, 0} 0; 1^*.$$

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.

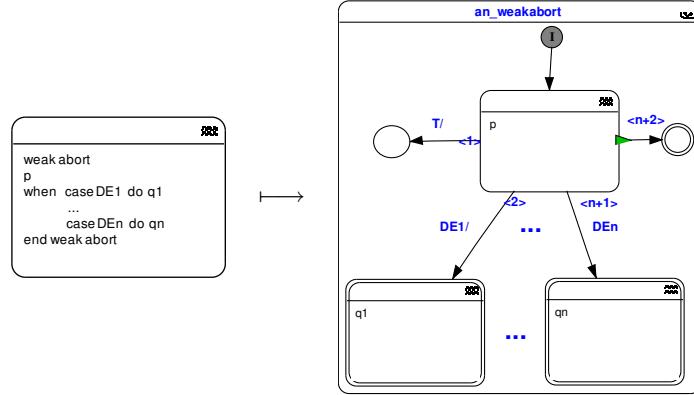
Beschreibung eines äquivalenten Makrozustandes

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** ausgelöst. Dabei ist das **T** eine Oder-Verknüpfung vom traphalt 3.4.24 Signal und den lokalen 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.

3. Transformation von Esterel in SyncCharts

Transformationsregel 28 (weakabort)



Transformationsregel 29 (weak abort in weak abort case)

weak abort p when DE \mapsto 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

3.5. Nicht transformierbare Befehle

$\langle SignalIdentifier \rangle$. 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.

3. Transformation von Esterel in SyncCharts

4. Optimierung von SyncCharts

In diesem Kapitel werden die verschiedenen 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 entstehen, 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.

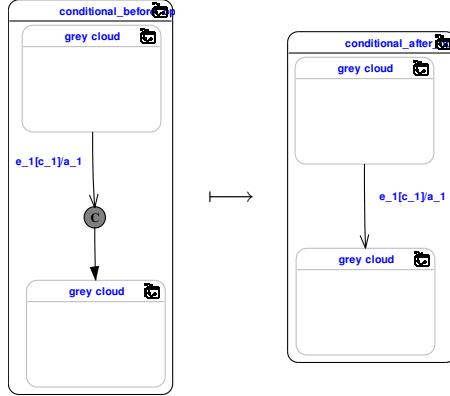
In den Regeln sind die Transitionsbeschriftungen in der Form $e[c]/a$ angegeben, wobei Indizierungen vorkommen. Dabei steht e für einen beliebigen *Trigger*, c für eine beliebige Kondition und a für beliebige Aktionen. Am Ende des Kapitels wird ein Algorithmus 4.1 angegeben, der einen Makrozustand optimiert. 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 Transition ohne 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*)

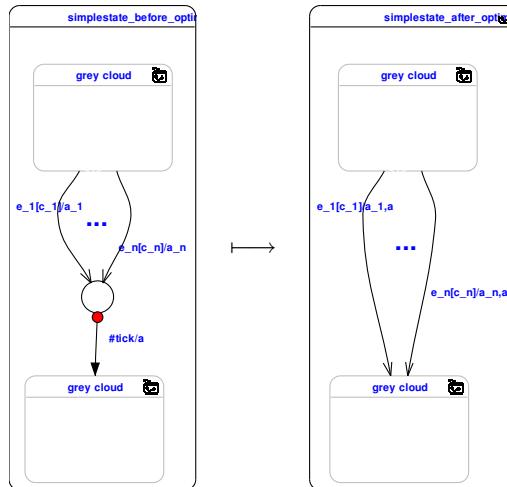
4. Optimierung von SyncCharts



4.0.2. Entfernen unnötiger, einfacher Zustände

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

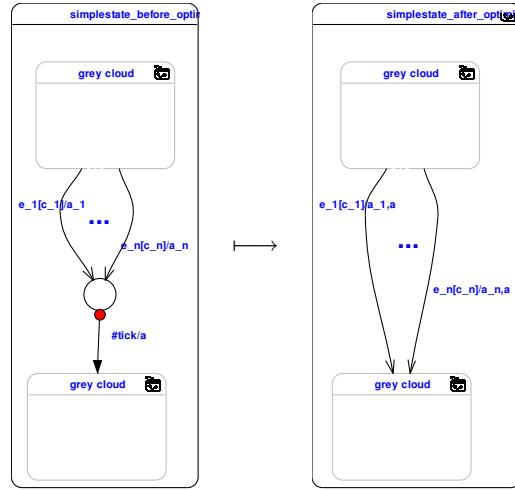
Optimierungsregel 2 (Entfernen unnötiger, einfacher Zustände 1)



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 $\langle \text{factor} \rangle$ addiert, und als Summe der eingehenden

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

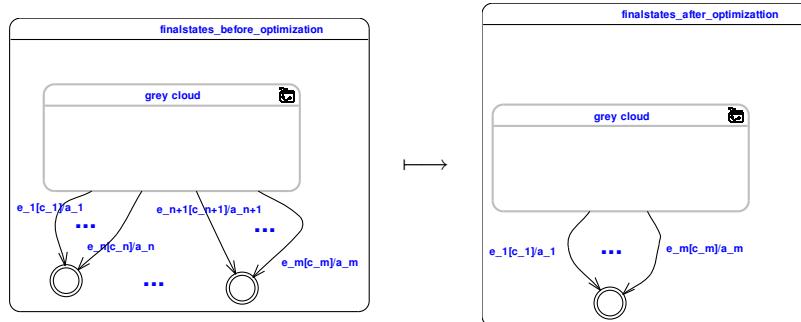
Optimierungsregel 3 (Entfernen unnötiger, einfacher Zustände 2)



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)

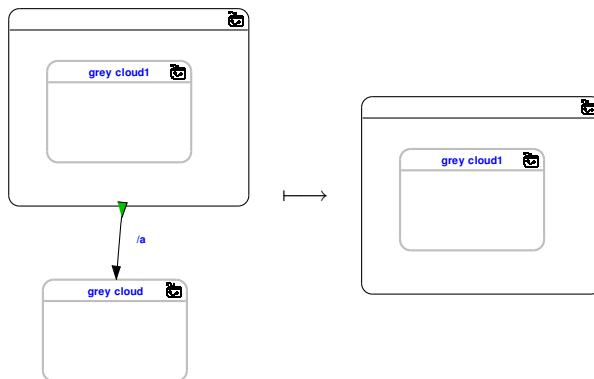


4. Optimierung von SyncCharts

4.0.4. Entfernen unnötiger *normal terminations*

Besitzt ein Makrozustand keine finalen Zustände, so wird eine vorhandene 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

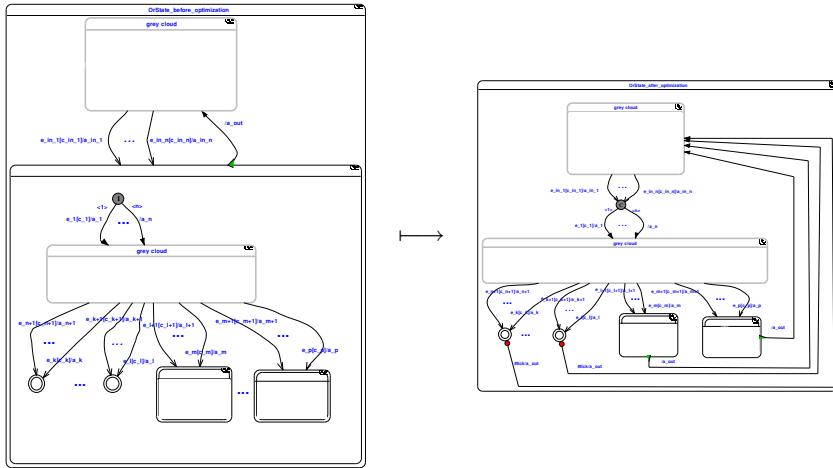
Um einen Makrozustand zu entfernen, müssen die folgenden Bedingungen erfüllt sein:

- 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 sichergestellt. Der Makrozustand mit seiner ausgehenden Transition kann 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.

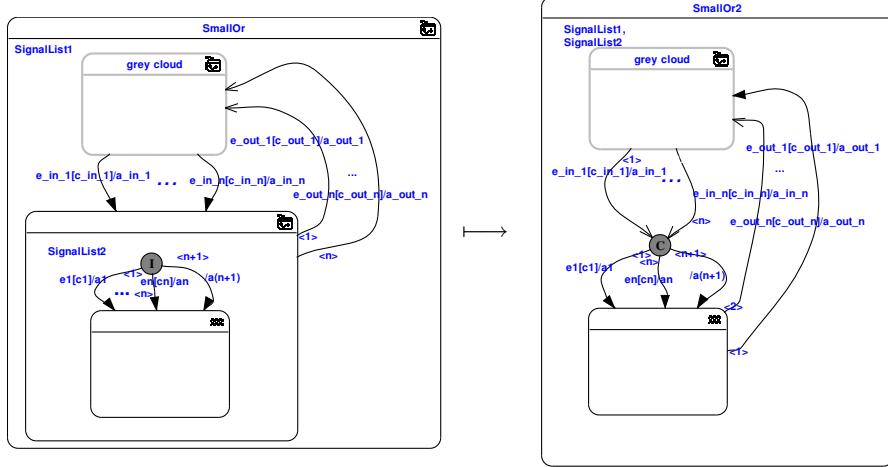
Der Vorgang verläuft analog zu dem in der Regel 4.0.5 beschriebenen Vorgehen. Zusätzlich werden eventuelle ausgehende *weak abortions* und *strong abortions* zu den ausgehenden Transitionen des Zustandes hinzugefügt und die lokalen Deklarationen dem übergeordneten Makrozustand hinzugefügt.

Optimierungsregel 7 (Entfernen von Makrozuständen mit nur zwei Unterzuständen)

4.0.7. Überprüfen auf finalen Charakter

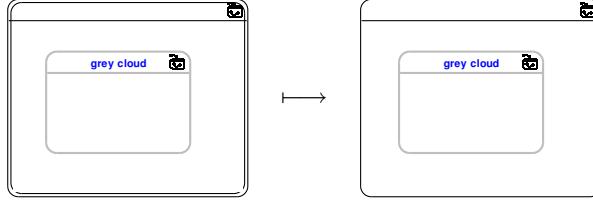
Besitzt ein finaler Makrozustand, keine untergeordneten finalen Zustände, so wird der Makrozustand in einen entsprechenden nicht-finalen Zustand umgewandelt. Eventu-

4. Optimierung von SyncCharts



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.

4.1. Der Algorithmus zum Optimieren eines Makrozustandes

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.

4. Optimierung von SyncCharts

5. Transformation am Beispiel 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 erschien 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.

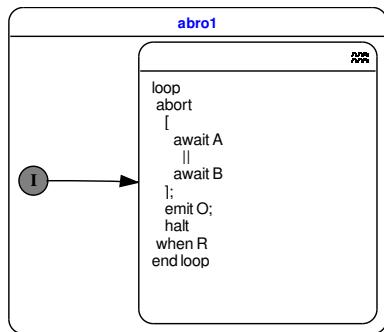
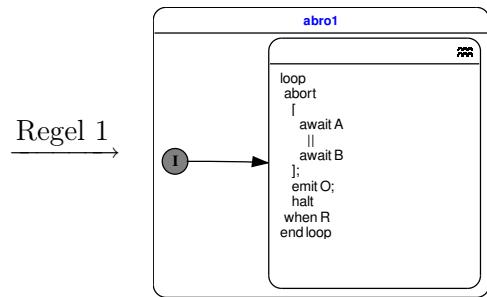
An dieser Stelle wird das Verhalten von *ABRO* vorgestellt. Zunächst wird auf die Signale A und B gewartet und anschließend das Signal O gesendet. Wird das Signal R empfangen, so wird der Vorgang abgebrochen und neu gestartet.

5.1. Beispielhafte Transformation von Esterel nach SyncChart

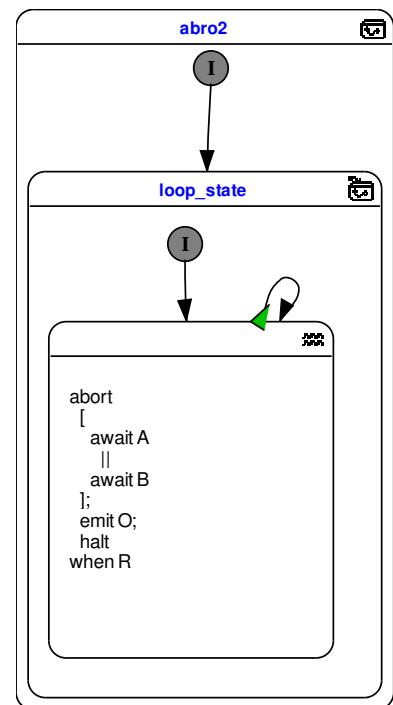
Das Programm *ABRO* wird folgendermaßen in ein SyncChart transformiert. Zuerst wird das `module` gemäß der Transformationsregel 1 umgewandelt. Dadurch werden insbesondere die Deklarationen in das SyncChart übernommen. Als nächstes wird `loop p end loop` mittels der Transformationsregel 17 umgewandelt. 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 parallelen 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.

5. Transformation am Beispiel von ABRO

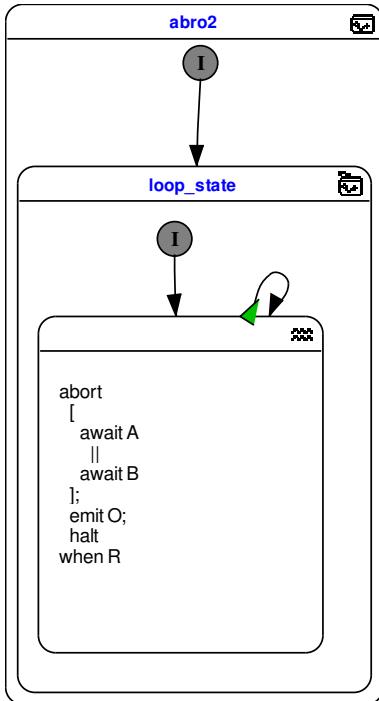
```
module abro:
input A,B,R;
output O;
loop
abort
[ await A
||| await B];
emit O;
halt
when R
end loop
end module
```



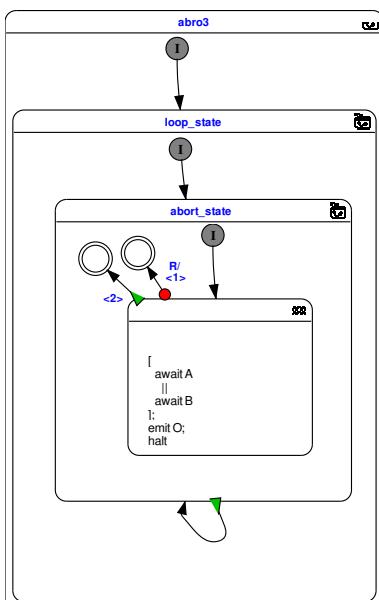
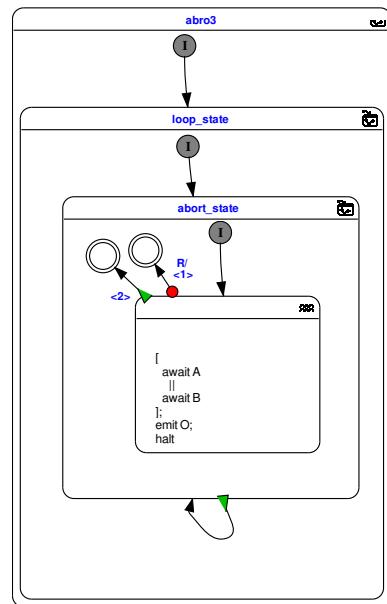
Regel 17



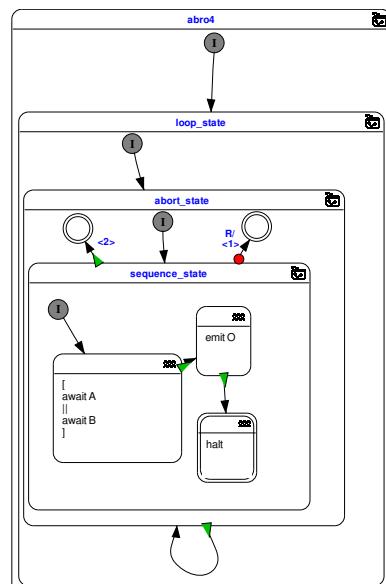
5.1. Beispielhafte Transformation von Esterel nach SyncChart



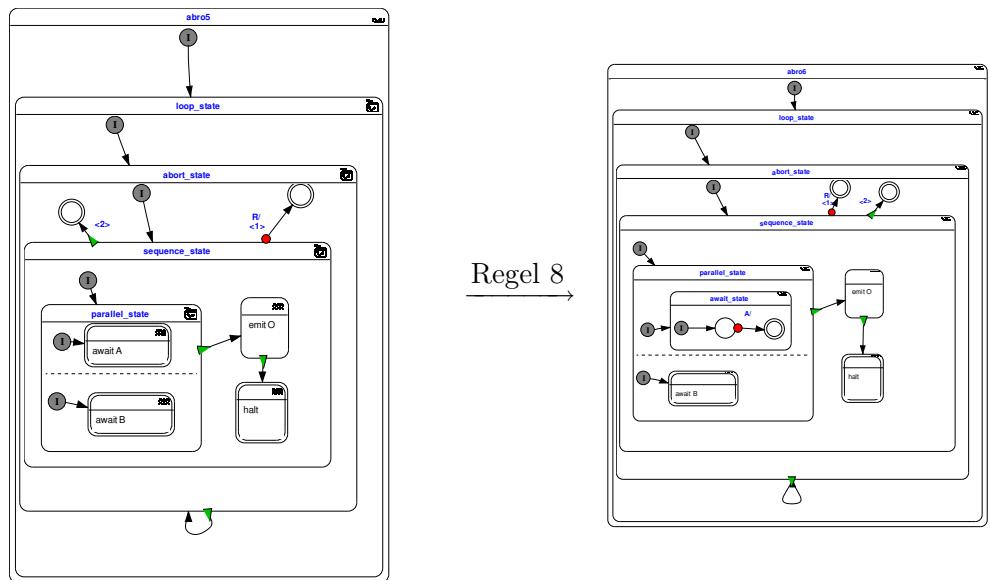
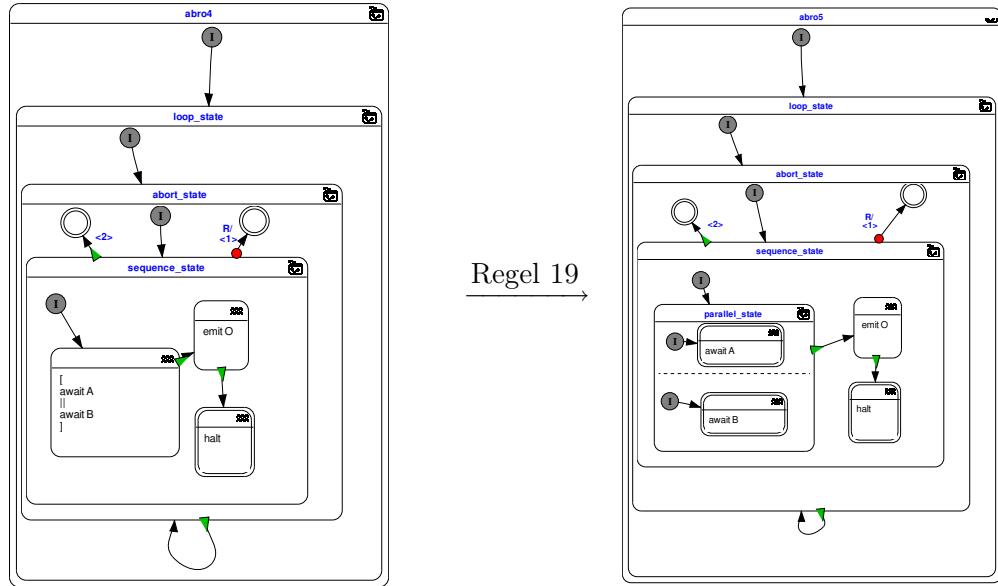
Regel 5 →



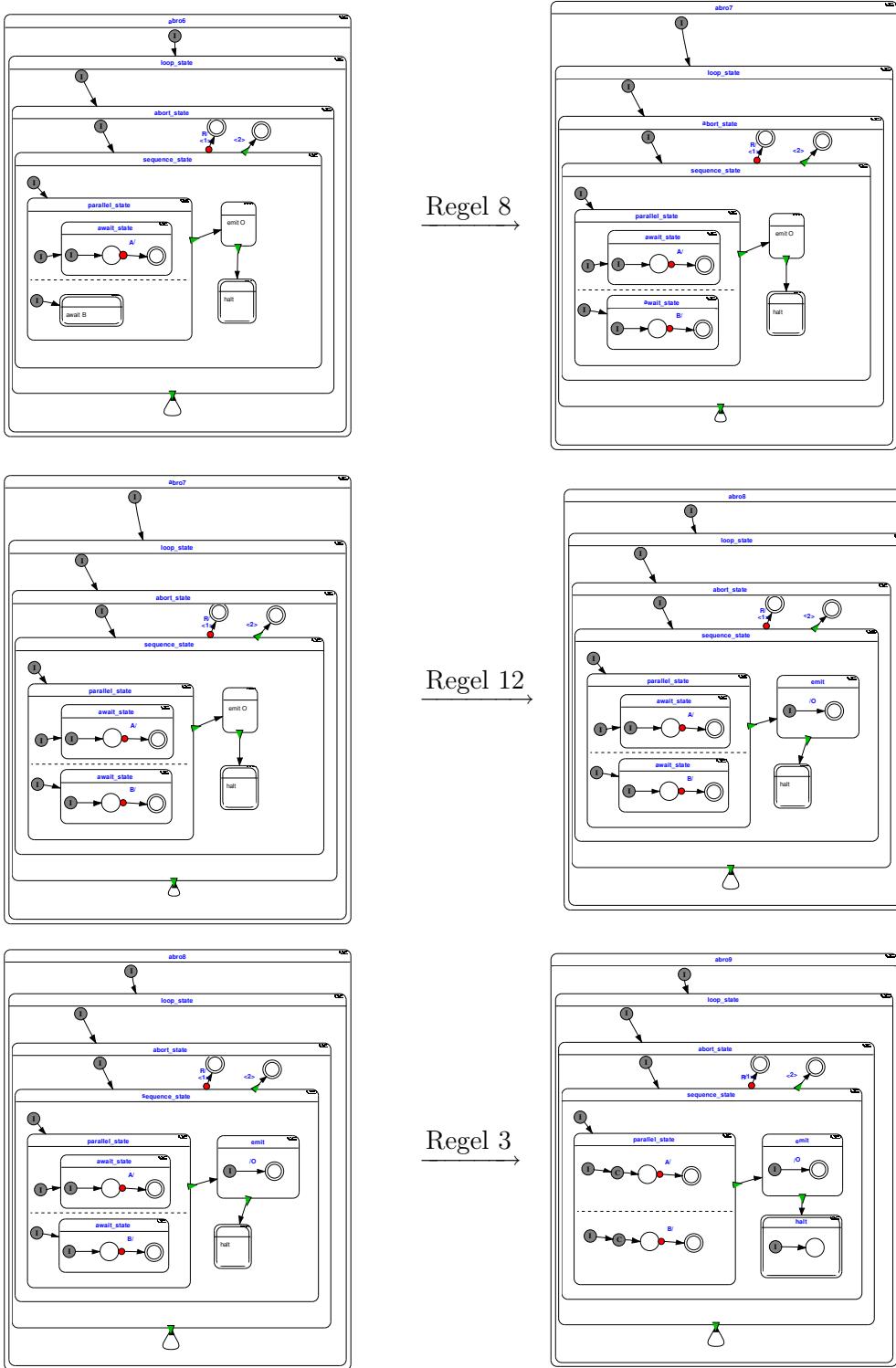
Regel 23 →



5. Transformation am Beispiel von ABRO



5.1. Beispielhafte Transformation von Esterel nach SyncChart

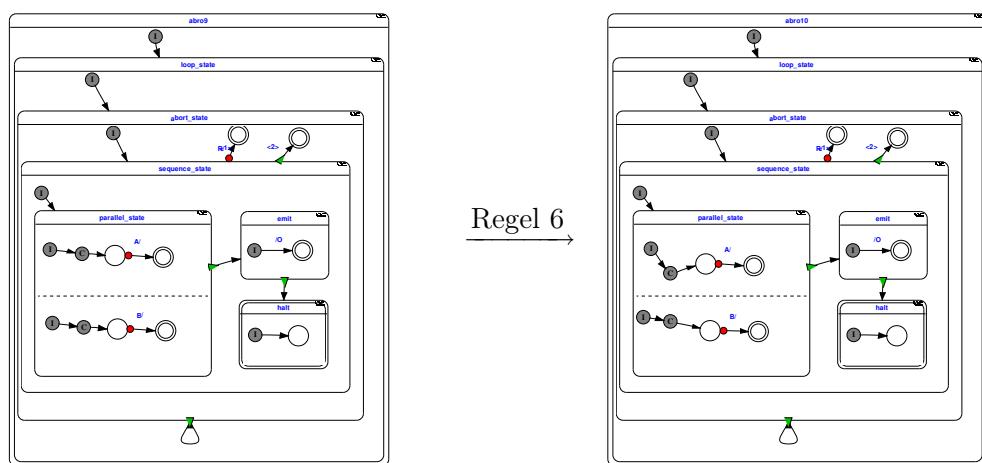


Es ist nun ein Statechart aus dem Esterel Programm entstanden, welches als nächstes Optimiert wird.

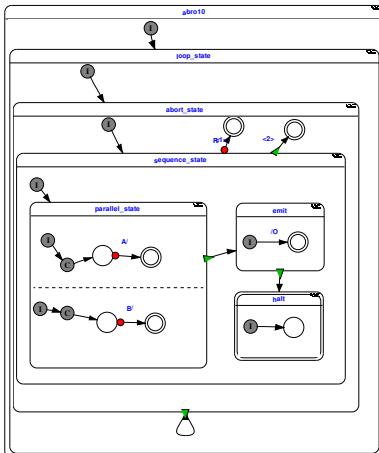
5. Transformation am Beispiel von ABRO

5.2. Optimierung

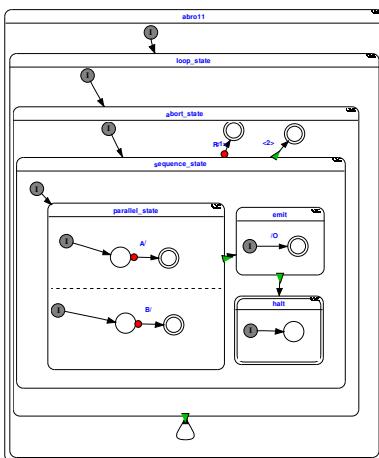
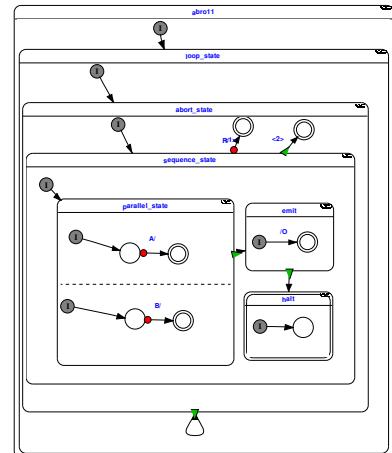
Zunächst werden die beiden parallelen Zustände mittels der Regel 6 nacheinander aufgelöst. Anschließend werden die gerade entstandenen *Conditional-Pseudozustände* durch Regel 1 aus dem SyncChart entfernt, und es werden die Zustände, welche aus `emit 0` und `halt` hervorgegangen sind, durch die Regel 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 entstandene *Conditional-Pseudozustand* durch die Regel 1 aus dem SyncChart entfernt. Das Ergebnis ist die optimale Darstellung des Esterel Programms *ABRO* als SyncChart.



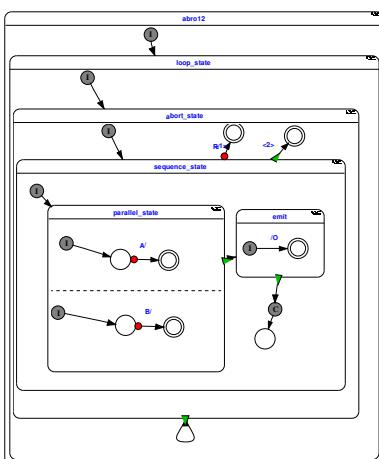
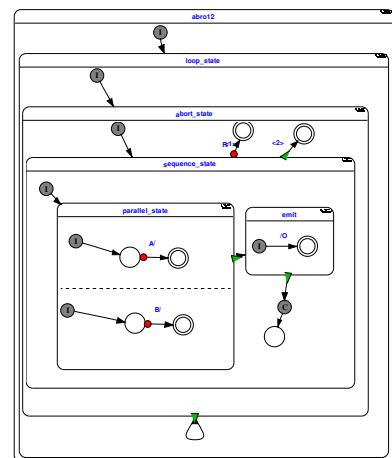
5.2. Optimierung



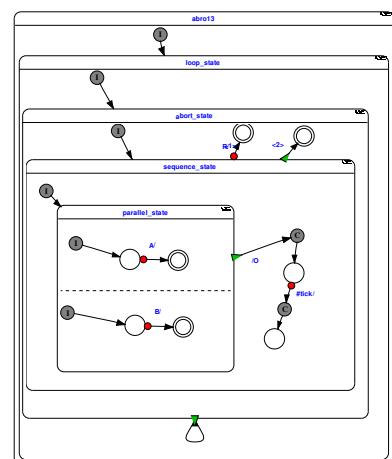
Regel 1 →



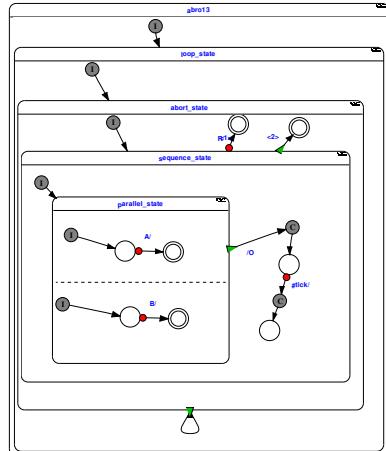
Regel 6 →



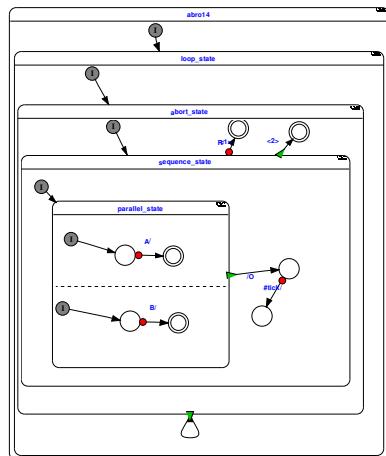
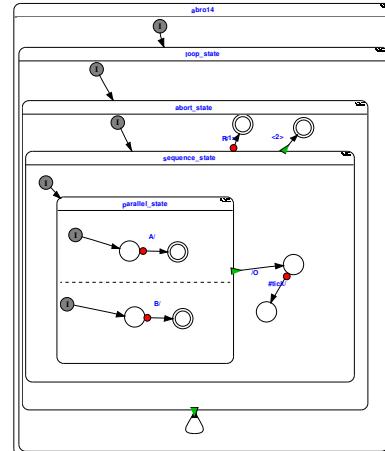
Regel 6 →



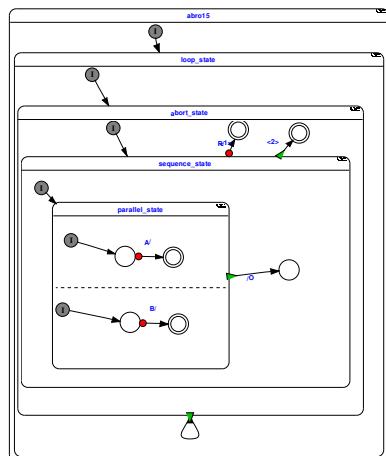
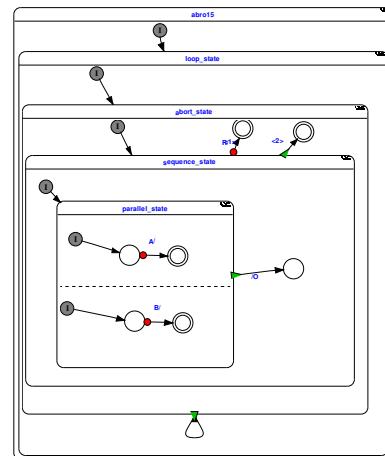
5. Transformation am Beispiel von ABRO



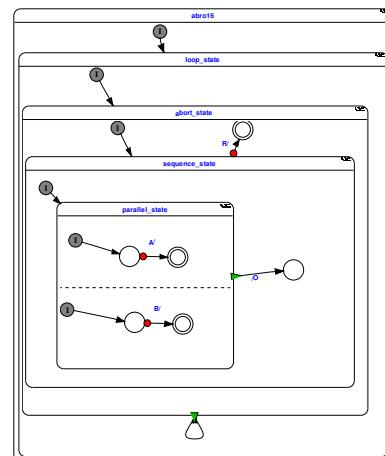
Regel 1 →



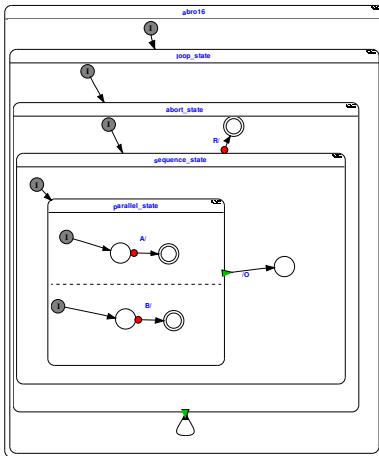
Regel 2 →



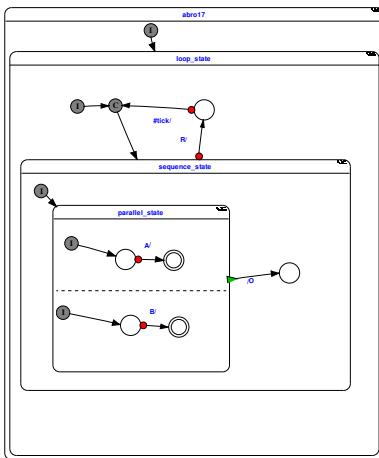
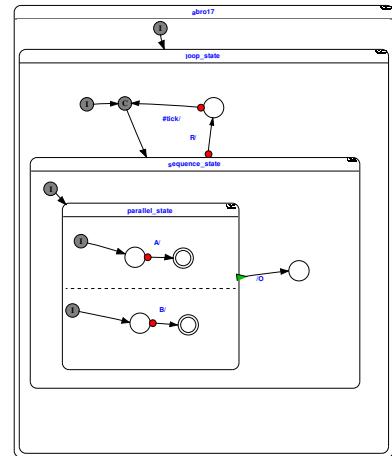
Regel 5 →



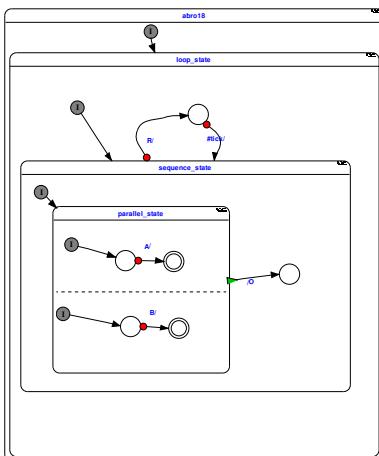
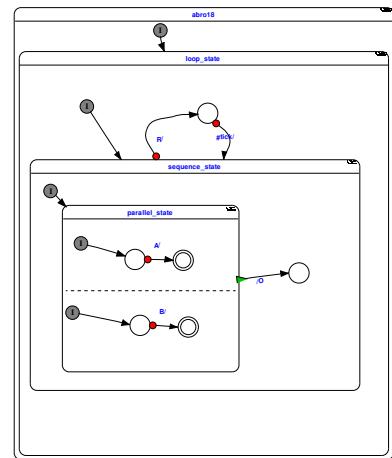
5.2. Optimierung



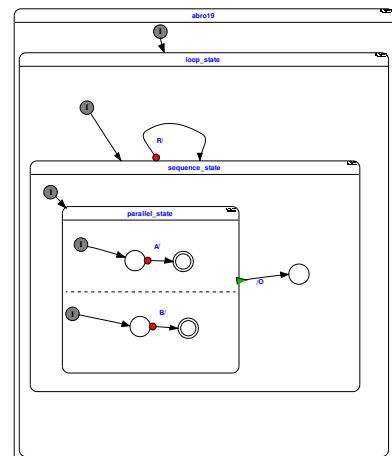
Regel 6 →



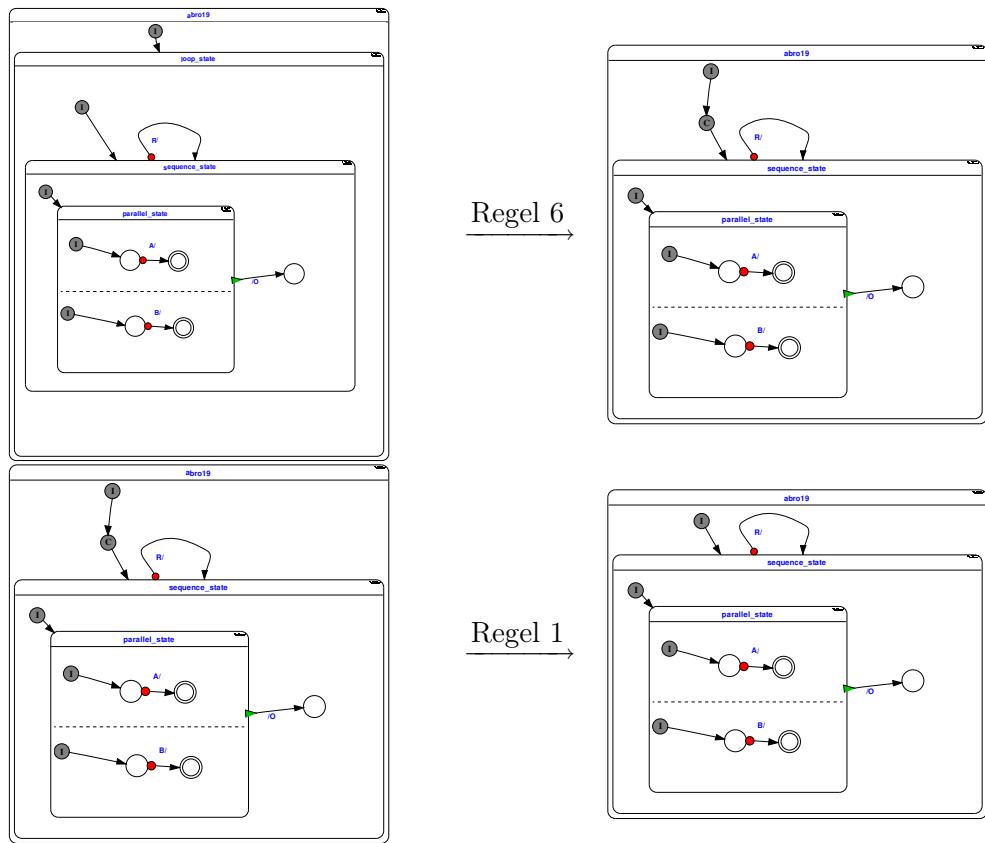
Regel 1 →



Regel 2 →



5. Transformation am Beispiel von ABRO



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 meistenverwendeten Schlüsselwörter kurz vor.

6. Die Implementation

//	:	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.

6.1. Die Java-Klassenstruktur für Esterel

ht

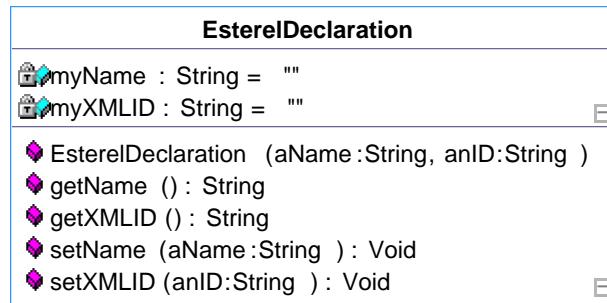


Abbildung 6.1.: UML-Darstellung der Klasse **EsterelDeclaration**

ht

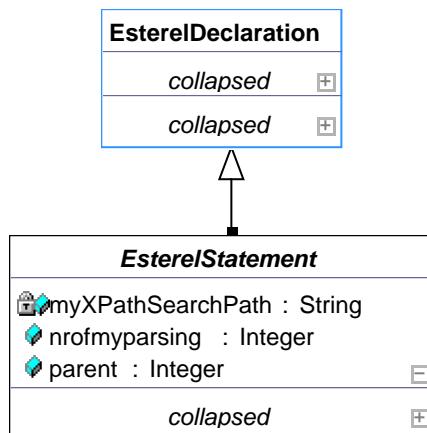


Abbildung 6.2.: UML-Darstellung der Klasse **EsterelStatement**

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

6. Die Implementation

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 Identifikationsnummer des XML-Elements besteht, zurück.
- `getXPathSearchString()` liefert den Wert des Klassenattributs zurück.
- `preParseEsterelStatement(EsterelParser)` setzt die Klassenattribute aus dem XML-Dokument.

Des Weiteren 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.

- `setSubStatement(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` beinhalten die lokalen Signale und lokalen Variablen, welche bekannt sind. Die letzte `ArrayList` enthält alle übergeordneten `TrapEsterelStatement` Objekte, um einen Zugriff auf die entsprechenden 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.

6.1. Die Java-Klassenstruktur für Esterel

ht

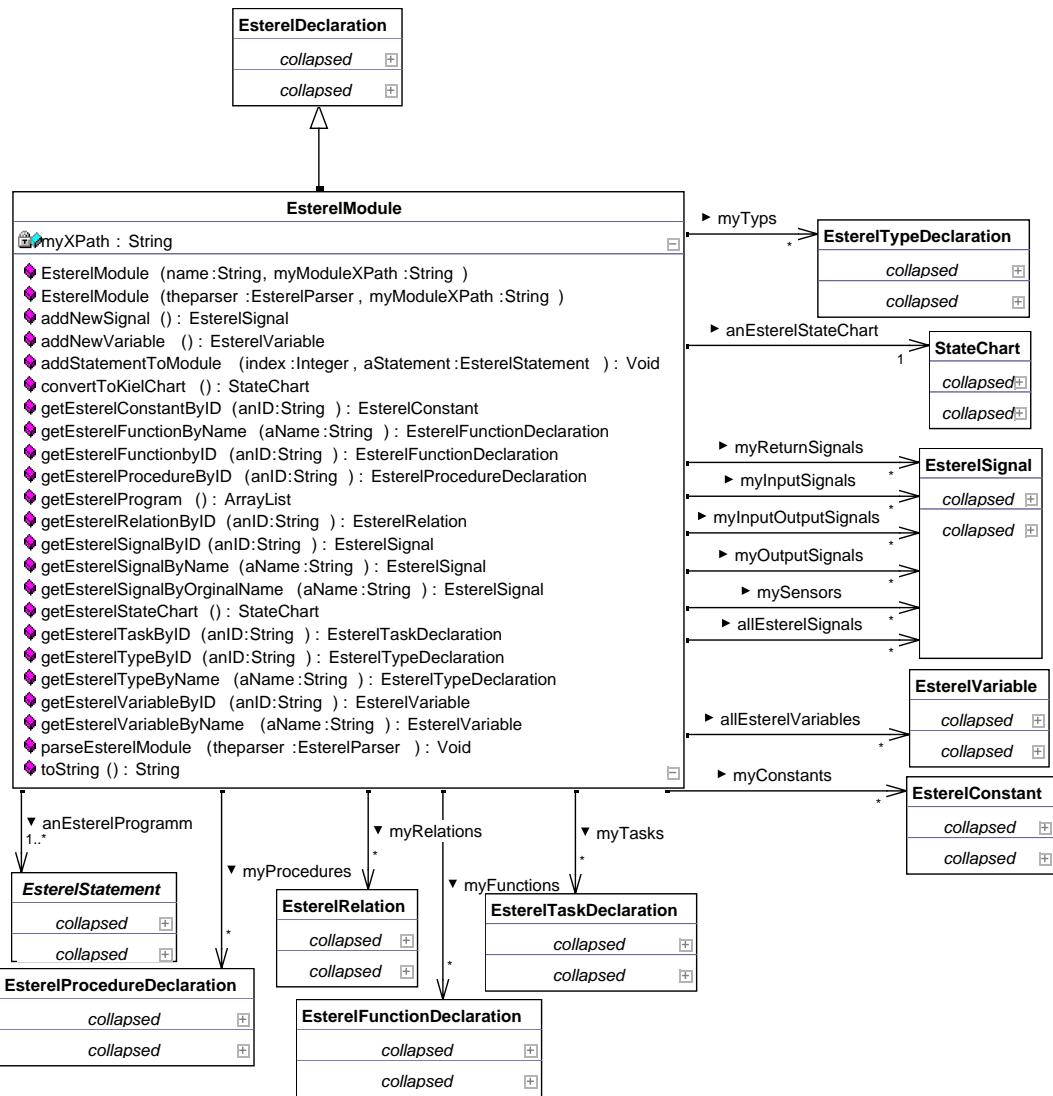


Abbildung 6.3.: UML-Darstellung der Klasse **EsterelModule**

6. Die Implementation

6.1.2. Die Klasse EsterelModule

Eine Instanz dieser Klasse enthält, wie in der Abbildung C.2.17 zu sehen, eine `ArrayList` namens `anEsterelProgramm`, die alle im Modul vorhandenen Befehle enthält. Dabei steht der Befehl, welcher dem Modul 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 Sync-Chart 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 `theElementXPath searchString` 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.

6.3. Die Transformation in die graphische Darstellung

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 `statementcounter` 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 Kapitel 3 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.

6. Die Implementation

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-Statements`, 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 verschiedenen Gründen Kompromisse bei der Transformation gemacht werden.

- Signale, die als *InputOutput* deklariert sind, werden in *KIEL* nur als Input Signal erstellt.
- Es können nur Konstanten 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 eigentlich 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 Komponenten 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.estrelstudio` jeweils als Klasse implementiert. Die Abbildung C.4 stellt die Klassenstruktur 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. Andernfalls 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 ebenweise 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 Makrozuständen des SyncCharts gibt, welches aufsteigend nach der Ebene der Makrozustände sortiert ist. Zudem bedarf es des Parameters `all` der angibt, ob schrittweise 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.

6. Die Implementation

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 Implementierungen 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 dieser Arbeit sind vielseitig und können sowohl als Dokumentation für Esterel-Programme dienen oder zur grafischen Simulation von Esterel-Programmen genutzt werden. Das wichtigste Anwendungsbereich 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 textliche Programmiersprache möglich wäre. Des Weiteren wäre es möglich, analog zu den hier entwickelten Transformationsregeln Transformationen in andere Statechart-Dialekte zu erstellen.

7. Zusammenfassung und Ausblick

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.estrel-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.de>. Christian-Albrechts-Universität zu Kiel 24098 Kiel, Germany.

A. Literaturverzeichnis

- [12] Columbia University. cec manpage. URL <http://www1.cs.columbia.edu/~sedwards/cec/cec.pdf>.
- [13] Duden. *Duden - Das große 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 Software Esterel Compiler. URL <http://www.esterel-technologies.com/technology/free-software/esterel-c%ompiler.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 Synccharts. 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).

A. Literaturverzeichnis

B. Bedienungsanleitung

B.1. Transformation von Esterel nach SyncChart

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

- *Java* muss zur Verfügung stehen,
- *KIEL* muss auf dem System vorhanden sein,
- der *cec* muss 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 beeinflusst werden, welche im Verzeichnis `/.kiel` vorhanden sind. Sollten man Veränderungen an diesen Dateien vorgenommen werden, kann es passieren dass 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*.

B. Bedienungsanleitung

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.

PreString Diese Zeichenkette wird am Anfang von jedem, bei der Umwandlung hinzugefügten, Signallnamen angefügt.

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

ArgumentSeparator 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 verschiedenen 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.

OptimizeESTEeliminateNeedlessConditional 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 Entfernen von unnötiger finaler Zuständen über `true`, wird `false` gesetzt, so wird die Regel deaktiviert.

OptimizeESTUpdateFinalStates aktiviert die Regel zum Umwandeln finaler in nicht-finales 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.

B. Bedienungsanleitung

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.estrel

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

C.1.1. EsterelFileFilter

Auflistung C.1: Die Klasse EsterelFileFilter

C.1.2. Esterel

Auflistung C.2: Die Klasse Esterel

```

package kiel.fileInterface.estrel;
 * @param dest
 * @param bufSize
 * @param bufferSize
 * @param force
 * @throws IOException
 * an exception
 */
public static void copyFile(
    final File src,
    final File dest,
    final int bufferSize,
    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[bufferSize];
    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();
                }
            }
        }
    }
}

 * <p>
 * A Esterel Plugin for the generic fileinterface.
 * </p>
 * <p>
 * <b>Details on this special class:</b> this implements the Interface all
 * Fileformat Parser/generator must implement.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2004
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 * <p>
 * @author <a href="mailto:lk@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

```

C.1. Paket kiel.fileInterface.estrel

C. Java Code für die Transformation von Esterel in SyncCharts

```

160      */
161      public final FileFilter getStateChartFileFilter() {
162          return EsterelFileFilter.getInstance();
163      }
164
165      /**
166       * alayouter which layouts the statechart.
167       */
168      //private Layouter alayouter = null;
169
170      /**
171       * Returns true.
172       */
173      /**
174       * @return <code>true</code>
175       */
176      public final boolean canRead() {
177          // TODO Auto-generated method stub
178          return true;
179      }
180
181      /**
182       * Returns false.
183       */
184      /**
185       * @return <code>false</code>
186       */
187      public final boolean canWrite() {
188          return false;
189      }
190
191      /**
192       * Returns Esterel Plugin.
193       */
194      /**
195       * @return "Esterel Plugin"
196       */
197      public final String getName() {
198          // TODO Auto-generated method stub
199          return "Esterel Plugin";
200      }
201
202      /**
203       * Returns null.
204       */
205      /**
206       * @return <code>null</code>
207       */
208      /**
209       * @throws FileInterfaceException
210       *          not used
211       */
212      public final View getReadView()
213          throws FileInterfaceException {
214          View av = null;
215          return av;
216      }
217
218      /**
219       * Returns an instance of <code>OriginalEsterelFileFilter</code>
220       */
221      /**
222       * Returns an instance of <code>OriginalEsterelFileFilter</code>
223       */
224      /**
225       * @return an instance of <code>OriginalEsterelFileFilter</code>
226       */
227  
```

C.1. Paket kiel.fileInterface.estrel

```

if (os.compareTo("windows") == 0) {
    // Make it work under cygwin
    File dummy = new File(
        "dummy.str");
    String runCecCommand = "export PATH="
        + EstereProperties.getBinPathCygWin() + "$PATH\\"
        + patchOCC.replace("\\",
            "/") + "cec -keep exp " + "-logfile "
        + EstereProperties.getNameOfFile()
        + " dummy.str";
    File callScript = new File(
        "runcec.sh");
    File adummyc = new File(
        "dummy.c");
    File aExpdummy = new File(
        "dummy.exp");
    File aKielLog = new File(
        expDir
            + execName);
    File aLog = new File(
        EstereProperties.getNameOfFile());
    File aKielLog = new File(
        expDir
            + "startcec.bat");
    try {
        copyFile(f,
            dummy,
            abUFFER,
            true);
        if (callScript.exists()) {
            callScript.delete();
        }
        callScript.createNewFile();
        FileWriter fw = new FileWriter(
            callScript);
        fw.write(runCecCommand);
        fw.flush();
        fw.close();
        String execRunOec = "cd "
            + callScript.getAbsolutePath()
            .lastIndexOf(File.separator)
            + System.getProperty("line.separator")
            + EstereProperties.getCygwinPath()
            + File.separator + "bash "
            + callScript.getName();
        if (bat.exists()) {
            bat.delete();
        }
        bat.createNewFile();
        FileWriter batfw = new FileWriter(
            bat);
        batfw.write(execRunOec);
        batfw.flush();
    } catch (IOException e) {
        theexpfile = new File(
            "dummy.exp");
        theexpfile.delete();
        theexpfile.createNewFile();
        copyFile(f,
            dummy,
            abUFFER,
            true);
        if (callScript.exists()) {
            callScript.delete();
        }
        callScript.createNewFile();
        FileWriter fw = new FileWriter(
            callScript);
        fw.write(runCecCommand);
        fw.flush();
        fw.close();
        String run = pathToOCC
            + "cec --keep exp " + "D " + expDir +
            "--logfile " + expDir;
        Process p = Runtime.getRuntime().exec(run);
        p.waitFor();
        if (p.waitFor() != 0) {
            System.out.println("Error executing OCC command: " + p.getErrorStream().readLine());
        }
    }
}

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

132           expDir + expName);
                           + "cec.log for more informations.");
                }
            } //if str
        try {
            if (theexpfile != null
                && theexpfile.canRead()) {
                SAXBuilder builder = new SAXBuilder();
                builder.setExpandEntities(false);
                Document doc = builder.build(theexpfile.getAbsoluteFile());
                try {
                    EsterelParser aparser = null;
                    aparser = new EsterelParser(
                        doc);
                    anEsterelModule = aparser.gettheEsterelModule();
                } catch (EsterelParserException epe) {
                    //epe.printStackTrace();
                    theExceptions += epe.getMessage();
                }
                System.out.println(anEsterelModule[0].toString());
                + System.getProperty("line.separator");
                aparser = null;
            }
            if (EsterelProperties.getDebugEnabled())
                && anEsterelModule != null) {
                System.out.println(anEsterelModule[0].toString());
            }
            if (aparser != null) {
                if (anEsterelModule.length > 0) {
                    aStateChartOfAnEsterelfile =
                        anEsterelModule[0].convertToKielChart();
                }
            }
            catch (EsterelStudioException e) {
                e.printStackTrace();
                theExceptions += e.getMessage()
                    + System.getProperty("line.separator");
            }
        } //if
        if (f.canRead()
            && f.exists()
            && f.getPath().toLowerCase().endsWith(".strl")) {
            theexpfile.delete();
            thefile = new File(
                expDir
                + chame);
            if (theFile.exists())
                theFile.delete();
        }
        catch (JavaLangStackOverflowError e) {
            theExceptions += e.getMessage()
                + System.getProperty("line.separator");
        }
    } else {
        theExceptions += "The .exp file was not created."
        + System.setProperty("line.separator")
        + "Please check .kiel" + File.separator
    }
}
/* Not implemented yet.
 */
* @see kiel.fileInterface.FileInterface#writeStateChartDocument
* (kiel.datastructure.Statechart,
* kiel.graphicalInformations.View, java.io.File)
*/

```

```

440     public final File writeStateChartDocument(
441         final Statechart s,
442         final View v,
443         final File f) {
444         // TODO Auto-generated method stub
445         return null;
446     }
447
448     /**
449      * counts recursive the number of states .
450      * @param acState
451      * @return a CompositeState
452      */
453     private int checkNumberOfStates(
454         final CompositeState acState) {
455         ArrayList subnodes = acState.getSubnodes();
456         int result = subnodes.size();
457         for (int i = 0; i < subnodes.size(); i++) {
458             if (subnodes.get(i) instanceof CompositeState) {
459                 result += checkNumberOfStates((CompositeState) subnodes.get(i));
460             }
461         }
462         return result;
463     }

```

C.1.3. EsterelParserException

Auflistung C.3: Die Klasse EsterelParserException

```

package kiel.fileInterface.esterel;
//import kiel.fileInterface.FileInterfaceException;
import javax.swing.JOptionPane;
/** 
 * <p> The Exception class for the EsterelParser. </p>
 * <p>Copyright: Copyright (c) 2005</p>
 * <p>Company: Uni Kiel</p>
 */
public class EsterelParserException extends /*FileInterface*/Exception {
    /**
     * the parser state in the moment of the exception.
     */
    private EsterelParser myEsterelParser;
    /**
     * the standard constructor.
     */
    public EsterelParserException() {
        super();
        this.myEsterelParser = null;
    }
    /**
     * sets error message.
     * @param arg0 the text
     */
    public EsterelParserException(final String arg0) {
        super(arg0);
    }
    /**
     * sets error message and parser state.
     * @param arg0 the error message
     * @param parser the parser
     * @see kiel.fileInterface.esterel.EsterelParser
     */
    public EsterelParserException(
        final String arg0,
        final EsterelParser parser) {
        this(arg0);
        this.myEsterelParser = parser;
    }
    /**
     * @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

C. Java Code für die Transformation von Esterel in SyncCharts

```

 * 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
 *
 * @see kiel.fileinterface.esterel.esterel2estudio.EsterelStatement
 *      is only delivered
 */
public final EsterelStatement createProgram() throws EsterelParserException {
    this.theElementXPathSearchString = /*[local-name(.)='Module']/*;
    List anElementList = DOMHelper.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 createProgram()

/**
 * <p>
 * Returns the <code>EsterelStatement</code> of the actual parsed
 * </p>
 * <br>
 * Sideeffects: The class variable <code>theElement</code> is set
 * @return <code>EsterelStatement</code>.
 * @param anElement
 * @throws EsterelParserException
 * @see kiel.fileinterface.esterel.esterel2estudio.EsterelStatement
 *      is only delivered
 * @see kiel.fileinterface.esterel.esterel2estudio.EsterelStatement
 *      is only delivered
 */
public final EsterelStatement createStatement(final Element anElement)
    throws EsterelParserException {
    String theElement = anElement.getName();
    // System.out.println(theElement);
    if (theElementStatement == null) {
        theElementStatement = new ParallelEsterelStatement(this);
        aReturnStatement = new AwaitEsterelStatement(this);
        else if (toIdentifyStatement == "Wait") {
            aReturnStatement = new NewEsterelStatement(this);
        } else if (toIdentifyStatement == "Loop") {
            aReturnStatement = new LoopEsterelStatement(this);
        } else if (toIdentifyStatement == "StatementList") {
            aReturnStatement = new StatementList();
        } else if (toIdentifyStatement == "SequenceEsterelStatement") {
            aReturnStatement = new SequenceEsterelStatement(this);
        } else if (toIdentifyStatement == "Inmt") {
            aReturnStatement = new InmtEsterelStatement(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 == "DoWatching") {
            aReturnStatement = new DoWatchingEsterelStatement(this);
        } else if (toIdentifyStatement == "Dobj") {
            aReturnStatement = new DobjEsterelStatement(this);
        } else {
            aReturnStatement = new DobjParserException("Unknown Statement.");
            + toIdentifyStatement, this);
        }
        if (aReturnStatement != null) {
            this.addData++;
        } else {
            throw new EsterelParserException("Unknown Statement.");
            + toIdentifyStatement, this);
        }
    }
    return aReturnStatement;
}

```

C.1. Paket kiel.fileInterface.estrel

```

    /**
     * Returns an index of the actual parsed estrel module. <br>
     * Sideeffects: none
     */
    220   * @return int
    public final int getActualEstrelModule() {
        return this.actualEstrelModule;
    } // public final int getActualEstrelModule()

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

    /**
     * Returns the actual parsed estrel module. <br>
     * Sideeffects: none
     */
    240   * @param i
    public final EstrelModule getElement(int i) {
        return this.theElementModule[i];
    } // public final EstrelModule getElement(int i)

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

    /**
     * Returns the class variable <code>statementCounter</code> by 1. <br>
     * Increments the class variable <code>statementCounter</code> by 1. <br>
     * Sideeffects: The class variable <code>statementCounter</code> is
     * increased by 1. <br>
     */
    260   * @return int
    public final int getLastParsedStatement() {
        return this.lastParsedStatement + 1;
    } // public final int getLastParsedStatement()
}

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

public final void incStatementCounter() {
    this.createStatementCounter++;
} // public final void incStatementCounter()

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

// public final void incStatementCounter(final int inc)

/** 
 * <p>
 * Parses a <code>org.jdom.Document</code> into the class variable
 * <code>theEsterelModule</code>.
 */
340   * Sideeffects: The class variable <code>anEsterelModule</code> is set.
 * The class variable <code>xmlDoc</code> is set with the parameter. <br>
 * The class variable <code>theElement</code> is used. <br>
 * 
 * @param anXmlDoc
 * the Document of an .exp(.xml) file
 * @throws EsterelParserException
 * @see kiel.fileinterface.estrel.estrel12studio.EsterelModule
 */
350   * @see kiel.fileinterface.estrel.estrel12studio.EsterelModule
 */

public final void parseExpFile(final Document xmlDoc)
    throws EsterelParserException {
    this.xmlDoc = xmlDoc;
    this.theElement = null;
    List allModules = DOMHelpers.getElements(
        "//*[Local-name(.)='ModuleSymbol']");
    if (allModules.size() < 0) {
        if (allModules.size() < 0) {
            throw new EsterelParserException("No module");
        }
    }
}

360   * 
 * @param allModules = new EsterelModule[allModules.size()];
 * set the names of the modules
 * for (this.actualEsterelModule = 0;
 */
370   * 
 * @return the string representation of a parser
 */
380   * Returns a string.
 */
390   * @return the string representation of a parser
 */
400   * 
 * @param statementCounter
 */
410   * 
 * @param statementCounter
 */
420   * 
 * @return the string representation of a parser
 */
430   * 
 * @return the string representation of a parser
 */
}

```

C.1.5. EsterelProperties

Auflistung C.5: Die Klasse EsterelProperties

```

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

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

/**
 * <p>
 * Used to load and store user definable properties.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * Company: Uni Kiel
 * </p>
 * Author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kettunen</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 cygwin bin path under windows.
     */
    private static final String CYGWINBINPATH = "cygwinBinPath";
    /**
     * The property key for debug mode.
     */
    private static final String DEBUGMODE = "DebugMode";
    /**
     * The default ressource for the properties.
     */
    private static final String DEFAULTRESOURCE = "estrel.properties";
}

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

/*
 * 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
    + "estrel.properties";

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

/*
 * path to the cec under windows.
 */
private static final String RELATIVEWINPATHTOCEC = "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(BINPATHINGCYGWIN);
}

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

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

140
160
170
180
190
200
210

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

    /**
     * @return the cygwinpath.
     */
    public static String getCygwinPath() {
        return properties.getProperty(CYGWINDIR);
    }

    /**
     * @return true, if debugnode is true.
     */
    public static boolean getDebugMode() {
        return properties.getProperty(DEBUGMODE)
            .equalsIgnoreCase("true");
    }

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

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

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

    /**
     * Loads the defaults from resource file.
     */
    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.
     */
    private static boolean loadProperties() {
        try {
            properties.load(new FileInputStream(
                PROPERTYFILE));
            * @return true, if loading was successful
        } catch (IOException e) {
            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.
     */
    protected static boolean load() {
        boolean success;
        Properties properties = new Properties(
            getDefaults());
        if (!loadProperties()) {
            success = copyDefaults();
        } else {
            success = true;
        }
        return success;
    }
}

```

```
        }
    }
    return success;
}
220 /**
 * Reloads the properties file.
 */
protected static void reload() {
    loadProperties();
}
230 /**
 * This is the constructor.
 */
private EstrelProperties() {
}
```

C. Java Code für die Transformation von Esterel in SyncCharts

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

Das Paket `kiel.fileInterface.estrel.estrel2estudio` enthält die Klassen zur Implementation der Transformationsregeln aus Kapitel 3. Zusätzlich werden die Klassen als UML-Darstellung angegeben.

C.2.1. AbortEstrelStatement

Der Klassenaufbau ist in der Abbildung C.2.1 dargestellt.

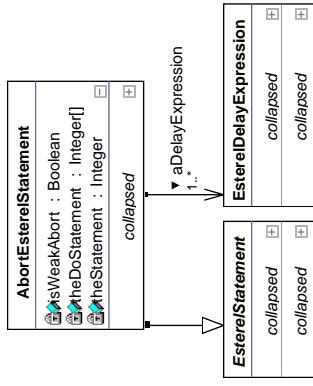


Abbildung C.1.: Klassendiagramm AbortEstrelStatement

Auflistung C.6: Die Klasse AbortEstrelStatement

```

package kiel.fileInterface.estrel.estrel2estudio;

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.WeakAbort;
import kiel.dataStructure.eventexp.DelayExpression;
import kiel.fileInterface.estrel.EstrelParser;
import kiel.util.DOMHelpers;
import kiel.util.StatechartHelpers;
import org.jdom.Element;

/*
 * Implements the estrel <>abort</code> statement.
 */
30  * <p>
 * <p>
 * Copyright : Copyright (c) 2005
 * <p>
 * <p>
 * Company : Uni Kiel
 * <p>
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
 * <br>
 * Version $Revision: 1.40 $ last modified $Date: 2006/02/06 18:35:28 $
40  *
*/
public class AbortEstrelStatement
    extends EstrelStatement {
    /**
     * The value of the Expressions. */
    private EstrelDelayExpression[] aDelayExpression;
    /**
     * If it is a weak abort or a strong abort.
     */
    private boolean isWeakAbort;
}

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

50      * The signal to halt an abort handler execution.
51      */
52      private EsterelSignal myHaltSignal = null;
53
54      /**
55       * the do statements.
56      */
57      private int[] theDoStatement;
58
59      /**
60       * the statement which to abort.
61      */
62      private int theStatement;
63
64      /**
65       * the standard constructor.
66      */
67      public AbortEsterelStatement() {
68          super();
69      }
70
71      /**
72       * Parses a Document and set the class variables. <br>
73       * It calls also the super constructor <code>EsterelStatement</code>EsterelStatement<(EsterelParser theparser)</code> .
74       * <br>
75       * Calls the <code>parseEsterelStatement</code> methode <br>
76       * Sideeffects: Changes <code>EsterelParser</code> class variables
77       *
78       * @param theparser
79       * @throws EsterelParserException
80       * @throws EsterelParserException
81       * @throws EsterelParserException
82       * @throws EsterelParserException
83       * @throws EsterelParserException
84       * @throws EsterelParserException
85       * @throws EsterelParserException
86       * @throws EsterelParserException
87       * @throws EsterelParserException
88       * @throws EsterelParserException
89       * @throws EsterelParserException
90       * @throws EsterelParserException
91       * @throws EsterelParserException
92       * @throws EsterelParserException
93       * @throws EsterelParserException
94       * @throws EsterelParserException
95       * @throws EsterelParserException
96       * @throws EsterelParserException
97       * @throws EsterelParserException
98       * @throws EsterelParserException
99       * @throws EsterelParserException
100      */
101      public AbortEsterelStatement(final EsterelParser theparser)
102          throws EsterelParserException {
103          super(
104              theparser);
105          if (theparser != null) {
106              this.parseEsterelStatement(theparser);
107              this.myHaltSignal =
108                  theparser.getEsterelModule(
109                      theparser.getActualEsterelModule());
110          }
111      }
112
113      /**
114       * Returns a state representation of the esterel statement.
115      */
116      /**
117       * the esterel module which becomes a KIEL Statechart
118      */
119      /**
120       * isFinalShare is true if the new state has to be a final state
121      */
122      /**
123       * theLocalEvents
124      */
125      /**
126       * theLocalVariables
127      */
128      /**
129       * theLocalVariables
130      */
131      /**
132       * theTreeTrapSignals
133      */
134      /**
135       * theTreeTrapSignals
136      */
137      /**
138       * theTreeTrapSignals
139      */
140      /**
141       * theTreeTrapSignals
142      */
143      /**
144       * theTreeTrapSignals
145      */
146      /**
147       * theTreeTrapSignals
148      */
149      /**
150       * theTreeTrapSignals
151      */
152
153      /**
154       * Converts the esterel module to Kiel
155      */
156      /**
157       * Converts the esterel module to Kiel
158      */
159      /**
160       * Converts the esterel module to Kiel
161      */

```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

C. Java Code für die Transformation von Esterel in SyncCharts

```

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

  /**
   * Implements the abstract methode from <code>EsterelStatement</code>
   * Parses an esterel statement in an <code>org.jdom.Document</code>
   * constructor <code> EsterelStatement(EsterelParser theparser)</code>
   * use the methode <code>preParseEsterelStatement</code> before
   * Sideeffects: the class variables are set Changes
   */
  public final void parseEsterelStatement(ESTERELPARSER theparser)
    throws EsterelParserException {
    List anElementList = DOMHelpers.getElements(theparser.getXMLDocument(),
      this.getPathsSearchString());
    theparser.incStatementCounter();
    if ((Element) anElementList.get(anElementList.size() - 1)).getTagName()
      .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++) {
      280
      if (((Element) anElementList.get(theCaseIndex)).getTagName()
        == "PredicatedStatement") {
        String aPredicatedStatementXPathSearchString =
          this.getPathsSearchString()
          + "[Local-name(.) ="
          + "([Element] anElementList.get(theCaseIndex)).getName()"
          + ", ][@id ="
          + "([Element] anElementList.get(theCaseIndex))";
        /**
         * Gets the attribute value of "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))";
        aPredicatedStatementXPathSearchString =
          ".getName()"
          + ", ][@id ="
          + "([Element] aPredicatedElementList.get(index))";
        /**
         * Gets the attribute value of "id"
         */
        + ",'']");
        this.aDelayExpression[theCaseIndex] =
          new EsterelDelayExpression[theCaseIndex];
        theparser,
        aSubPredicatedStatementXPathSearchString,
        theparser.getXMLDocument();
      } else {
        throw new EsterelParserException(
          this.getEsterelStatementName());
      }
    } //else if size/
  }
  /**
   * Fills the substatements in the esterel module. In this class it is
   * fills the do Statements if there are any <br>
   * Sideffects: none
   * @param theparser the actual parser
   * @throws EsterelParserException
   * @see kiel.fileInterface.estrel.EsterelParserException
   * @see kiel.fileInterface.estrel.EsterelParser
   */
  public final void parseEsterelStatement(final EsterelParser
    320
    330
    340
    350
    360
    370
    380
    390
    400
    410
    420
    430
    440
    450
    460
    470
    480
    490
    500
    510
    520
    530
    540
    550
    560
    570
    580
    590
    600
    610
    620
    630
    640
    650
    660
    670
    680
    690
    700
    710
    720
    730
    740
    750
    760
    770
    780
    790
    800
    810
    820
    830
    840
    850
    860
    870
    880
    890
    900
    910
    920
    930
    940
    950
    960
    970
    980
    990
    1000
    1010
    1020
    1030
    1040
    1050
    1060
    1070
    1080
    1090
    1100
    1110
    1120
    1130
    1140
    1150
    1160
    1170
    1180
    1190
    1200
    1210
    1220
    1230
    1240
    1250
    1260
    1270
    1280
    1290
    1300
    1310
    1320
    1330
    1340
    1350
    1360
    1370
    1380
    1390
    1400
    1410
    1420
    1430
    1440
    1450
    1460
    1470
    1480
    1490
    1500
    1510
    1520
    1530
    1540
    1550
    1560
    1570
    1580
    1590
    1600
    1610
    1620
    1630
    1640
    1650
    1660
    1670
    1680
    1690
    1700
    1710
    1720
    1730
    1740
    1750
    1760
    1770
    1780
    1790
    1800
    1810
    1820
    1830
    1840
    1850
    1860
    1870
    1880
    1890
    1900
    1910
    1920
    1930
    1940
    1950
    1960
    1970
    1980
    1990
    2000
    2010
    2020
    2030
    2040
    2050
    2060
    2070
    2080
    2090
    2100
    2110
    2120
    2130
    2140
    2150
    2160
    2170
    2180
    2190
    2200
    2210
    2220
    2230
    2240
    2250
    2260
    2270
    2280
    2290
    2300
    2310
    2320
    2330
    2340
    2350
    2360
    2370
    2380
    2390
    2400
    2410
    2420
    2430
    2440
    2450
    2460
    2470
    2480
    2490
    2500
    2510
    2520
    2530
    2540
    2550
    2560
    2570
    2580
    2590
    2600
    2610
    2620
    2630
    2640
    2650
    2660
    2670
    2680
    2690
    2700
    2710
    2720
    2730
    2740
    2750
    2760
    2770
    2780
    2790
    2800
    2810
    2820
    2830
    2840
    2850
    2860
    2870
    2880
    2890
    2900
    2910
    2920
    2930
    2940
    2950
    2960
    2970
    2980
    2990
    3000
    3010
    3020
    3030
    3040
    3050
    3060
    3070
    3080
    3090
    3100
    3110
    3120
    3130
    3140
    3150
    3160
    3170
    3180
    3190
    3200
    3210
    3220
    3230
    3240
    3250
    3260
    3270
    3280
    3290
    3300
    3310
    3320
    3330
    3340
    3350
    3360
    3370
    3380
    3390
    3400
    3410
    3420
    3430
    3440
    3450
    3460
    3470
    3480
    3490
    3500
    3510
    3520
    3530
    3540
    3550
    3560
    3570
    3580
    3590
    3600
    3610
    3620
    3630
    3640
    3650
    3660
    3670
    3680
    3690
    3700
    3710
    3720
    3730
    3740
    3750
    3760
    3770
    3780
    3790
    3800
    3810
    3820
    3830
    3840
    3850
    3860
    3870
    3880
    3890
    3900
    3910
    3920
    3930
    3940
    3950
    3960
    3970
    3980
    3990
    4000
    4010
    4020
    4030
    4040
    4050
    4060
    4070
    4080
    4090
    4100
    4110
    4120
    4130
    4140
    4150
    4160
    4170
    4180
    4190
    4200
    4210
    4220
    4230
    4240
    4250
    4260
    4270
    4280
    4290
    4300
    4310
    4320
    4330
    4340
    4350
    4360
    4370
    4380
    4390
    4400
    4410
    4420
    4430
    4440
    4450
    4460
    4470
    4480
    4490
    4500
    4510
    4520
    4530
    4540
    4550
    4560
    4570
    4580
    4590
    4600
    4610
    4620
    4630
    4640
    4650
    4660
    4670
    4680
    4690
    4700
    4710
    4720
    4730
    4740
    4750
    4760
    4770
    4780
    4790
    4800
    4810
    4820
    4830
    4840
    4850
    4860
    4870
    4880
    4890
    4900
    4910
    4920
    4930
    4940
    4950
    4960
    4970
    4980
    4990
    5000
    5010
    5020
    5030
    5040
    5050
    5060
    5070
    5080
    5090
    5100
    5110
    5120
    5130
    5140
    5150
    5160
    5170
    5180
    5190
    5200
    5210
    5220
    5230
    5240
    5250
    5260
    5270
    5280
    5290
    5300
    5310
    5320
    5330
    5340
    5350
    5360
    5370
    5380
    5390
    5400
    5410
    5420
    5430
    5440
    5450
    5460
    5470
    5480
    5490
    5500
    5510
    5520
    5530
    5540
    5550
    5560
    5570
    5580
    5590
    5600
    5610
    5620
    5630
    5640
    5650
    5660
    5670
    5680
    5690
    5700
    5710
    5720
    5730
    5740
    5750
    5760
    5770
    5780
    5790
    5800
    5810
    5820
    5830
    5840
    5850
    5860
    5870
    5880
    5890
    5900
    5910
    5920
    5930
    5940
    5950
    5960
    5970
    5980
    5990
    6000
    6010
    6020
    6030
    6040
    6050
    6060
    6070
    6080
    6090
    6100
    6110
    6120
    6130
    6140
    6150
    6160
    6170
    6180
    6190
    6200
    6210
    6220
    6230
    6240
    6250
    6260
    6270
    6280
    6290
    6300
    6310
    6320
    6330
    6340
    6350
    6360
    6370
    6380
    6390
    6400
    6410
    6420
    6430
    6440
    6450
    6460
    6470
    6480
    6490
    6500
    6510
    6520
    6530
    6540
    6550
    6560
    6570
    6580
    6590
    6600
    6610
    6620
    6630
    6640
    6650
    6660
    6670
    6680
    6690
    6700
    6710
    6720
    6730
    6740
    6750
    6760
    6770
    6780
    6790
    6800
    6810
    6820
    6830
    6840
    6850
    6860
    6870
    6880
    6890
    6900
    6910
    6920
    6930
    6940
    6950
    6960
    6970
    6980
    6990
    7000
    7010
    7020
    7030
    7040
    7050
    7060
    7070
    7080
    7090
    7100
    7110
    7120
    7130
    7140
    7150
    7160
    7170
    7180
    7190
    7200
    7210
    7220
    7230
    7240
    7250
    7260
    7270
    7280
    7290
    7300
    7310
    7320
    7330
    7340
    7350
    7360
    7370
    7380
    7390
    7400
    7410
    7420
    7430
    7440
    7450
    7460
    7470
    7480
    7490
    7500
    7510
    7520
    7530
    7540
    7550
    7560
    7570
    7580
    7590
    7600
    7610
    7620
    7630
    7640
    7650
    7660
    7670
    7680
    7690
    7700
    7710
    7720
    7730
    7740
    7750
    7760
    7770
    7780
    7790
    7800
    7810
    7820
    7830
    7840
    7850
    7860
    7870
    7880
    7890
    7900
    7910
    7920
    7930
    7940
    7950
    7960
    7970
    7980
    7990
    8000
    8010
    8020
    8030
    8040
    8050
    8060
    8070
    8080
    8090
    8100
    8110
    8120
    8130
    8140
    8150
    8160
    8170
    8180
    8190
    8200
    8210
    8220
    8230
    8240
    8250
    8260
    8270
    8280
    8290
    8300
    8310
    8320
    8330
    8340
    8350
    8360
    8370
    8380
    8390
    8400
    8410
    8420
    8430
    8440
    8450
    8460
    8470
    8480
    8490
    8500
    8510
    8520
    8530
    8540
    8550
    8560
    8570
    8580
    8590
    8600
    8610
    8620
    8630
    8640
    8650
    8660
    8670
    8680
    8690
    8700
    8710
    8720
    8730
    8740
    8750
    8760
    8770
    8780
    8790
    8800
    8810
    8820
    8830
    8840
    8850
    8860
    8870
    8880
    8890
    8900
    8910
    8920
    8930
    8940
    8950
    8960
    8970
    8980
    8990
    9000
    9010
    9020
    9030
    9040
    9050
    9060
    9070
    9080
    9090
    9100
    9110
    9120
    9130
    9140
    9150
    9160
    9170
    9180
    9190
    9200
    9210
    9220
    9230
    9240
    9250
    9260
    9270
    9280
    9290
    9300
    9310
    9320
    9330
    9340
    9350
    9360
    9370
    9380
    9390
    9400
    9410
    9420
    9430
    9440
    9450
    9460
    9470
    9480
    9490
    9500
    9510
    9520
    9530
    9540
    9550
    9560
    9570
    9580
    9590
    9600
    9610
    9620
    9630
    9640
    9650
    9660
    9670
    9680
    9690
    9700
    9710
    9720
    9730
    9740
    9750
    9760
    9770
    9780
    9790
    9800
    9810
    9820
    9830
    9840
    9850
    9860
    9870
    9880
    9890
    9900
    9910
    9920
    9930
    9940
    9950
    9960
    9970
    9980
    9990
    9999
    10000
    10001
    10002
    10003
    10004
    10005
    10006
    10007
    10008
    10009
    10010
    10011
    10012
    10013
    10014
    10015
    10016
    10017
    10018
    10019
    10020
    10021
    10022
    10023
    10024
    10025
    10026
    10027
    10028
    10029
    10030
    10031
    10032
    10033
    10034
    10035
    10036
    10037
    10038
    10039
    10040
    10041
    10042
    10043
    10044
    10045
    10046
    10047
    10048
    10049
    10050
    10051
    10052
    10053
    10054
    10055
    10056
    10057
    10058
    10059
    10060
    10061
    10062
    10063
    10064
    10065
    10066
    10067
    10068
    10069
    10070
    10071
    10072
    10073
    10074
    10075
    10076
    10077
    10078
    10079
    10080
    10081
    10082
    10083
    10084
    10085
    10086
    10087
    10088
    10089
    10090
    10091
    10092
    10093
    10094
    10095
    10096
    10097
    10098
    10099
    100100
    100101
    100102
    100103
    100104
    100105
    100106
    100107
    100108
    100109
    100110
    100111
    100112
    100113
    100114
    100115
    100116
    100117
    100118
    100119
    100120
    100121
    100122
    100123
    100124
    100125
    100126
    100127
    100128
    100129
    100130
    100131
    100132
    100133
    100134
    100135
    100136
    100137
    100138
    100139
    100140
    100141
    100142
    100143
    100144
    100145
    100146
    100147
    100148
    100149
    100150
    100151
    100152
    100153
    100154
    100155
    100156
    100157
    100158
    100159
    100160
    100161
    100162
    100163
    100164
    100165
    100166
    100167
    100168
    100169
    100170
    100171
    100172
    100173
    100174
    100175
    100176
    100177
    100178
    100179
    100180
    100181
    100182
    100183
    100184
    100185
    100186
    100187
    100188
    100189
    100190
    100191
    100192
    100193
    100194
    100195
    100196
    100197
    100198
    100199
    100200
    100201
    100202
    100203
    100204
    100205
    100206
    100207
    100208
    100209
    100210
    100211
    100212
    100213
    100214
    100215
    100216
    100217
    100218
    100219
    100220
    100221
    100222
    100223
    100224
    100225
    100226
    100227
    100228
    100229
    100230
    100231
    100232
    100233
    100234
    100235
    100236
    100237
    100238
    100239
    100240
    100241
    100242
    100243
    100244
    100245
    100246
    100247
    100248
    100249
    100250
    100251
    100252
    100253
    100254
    100255
    100256
    100257
    100258
    100259
    100260
    100261
    100262
    100263
    100264
    100265
    100266
    100267
    100268
    100269
    100270
    100271
    100272
    100273
    100274
    100275
    100276
    100277
    100278
    100279
    100280
    100281
    100282
    100283
    100284
    100285
    100286
    100287
    100288
    100289
    100290
    100291
    100292
    100293
    100294
    100295
    100296
    100297
    100298
    100299
    100300
    100301
    100302
    100303
    100304
    100305
    100306
    100307
    100308
    100309
    100310
    100311
    100312
    100313
    100314
    100315
    100316
    100317
    100318
    100319
    100320
    100321
    100322
    100323
    100324
    100325
    100326
    100327
    100328
    100329
    100330
    100331
    100332
    100333
    100334
    100335
    100336
    100337
    100338
    100339
    100340
    100341
    100342
    100343
    100344
    100345
    100346
    100347
    100348
    100349
    100350
    100351
    100352
    100353
    100354
    100355
    100356
    100357
    100358
    100359
    100360
    100361
    100362
    100363
    100364
    100365
    100366
    100367
    100368
    100369
    100370
    100371
    100372
    100373
    100374
    100375
    100376
    100377
    100378
    100379
    100380
    100381
    100382
    100383
    100384
    100385
    100386
    100387
    100388
    100389
    100390
    100391
    100392
    100393
    100394
    100395
    100396
    100397
    100398
    100399
    100400
    100401
    10
```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

    * @param anEstrelModule
    *   an estrel modul representation
    * @return a String representation of an estrel statement
    * @see kiel.fileInterface.estrel.estrel2estudio.EstrelStatement
    */
    public final String toString(
        final EstrelModule anEstrelModule) {
        final String result = "";
        if (this.isWeakAbort) {
            result += "weak ";
        }
        result += this.getEstrelStatementName();
        result += "\n";
        result += ((EstrelStatement)
            (anEstrelModule
                .getEstrelModule()
                .getEstrelProgram()
                .getEstrelIPProgram()
                .get(this.getStatement())));
        result += "\n when ";
        if (this.isDelayExpression.length > 1) {
            for (int i = 0;
                theCaseIndex < this.delayExpression.length;
                i++) {
                theCaseIndex++;
                result += "n";
            }
        }
        result += " delayExpression[" + theCaseIndex + "]";
        result += " ";
        if (theCaseIndex < this.delayExpression.length) {
            theCaseIndex++;
            result += "n";
        }
        result += " case ";
        result += this.delayExpression[theCaseIndex];
        result += "\n end ";
        result += "\n";
    }
    */
    /**
     * Implements the abstract methode from <code>EstrelStatement</code>.
     * Returns a string representation of an estrel statement.
     */
    public void addStatementToModule(TheParser parser,
        final EstrelModule module) {
        final ElementList elementList =
            module.createElementList();
        final String str =
            this.getEstrelStatementName() +
            ((EstrelStatement)
                (module
                    .getEstrelModule()
                    .getEstrelProgram()
                    .getEstrelIPProgram()
                    .get(this.getStatement())));
        elementList.add(str);
    }
}

```

C.2.2. AssignEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.2 dargestellt.

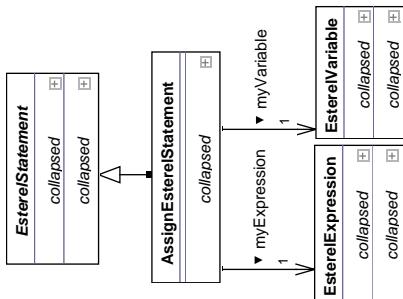


Abbildung C.2.: Klassendiagramm AssignEsterelStatement

Auflistung C.7: Die Klasse AssignEsterelStatement

```

20   * <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>
     *
     * <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
     * @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.
         */
    }
  
```

10

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

40    public AssignEsterelStatement() {
41        super();
42        this.myExpression = null;
43        this.myVariable = null;
44    }
45    /**
46     * Parses a Document and set the class variables.
47     * cbrIt calls also the super constructor<code>EsterelStatement
48     * </code>.
49     * @param theparser theparser
50     */
51    public AssignEsterelStatement(ESTERELParser theparser) {
52        * Changes <code>EsterelParser</code> class variables
53        * @param theparser an intace of a EsterelParser
54        * Throws EsterelParserException if the parser is only delivered
55        * @see kiel.fileinterface.estrel.EsterelParserException
56        * @see EsterelStatement(EsterelParser)
57        * @see EsterelStatement(EsterelParser theparser)
58        * @see kiel.fileinterface.estrel.EsterelParser
59        */
60        public AssignEsterelStatement(final EsterelParser theparser)
61            throws EsterelParserException {
62            super(theparser);
63            this.myExpression = null;
64            this.myVariable = null;
65            if (theparser != null) {
66                this.parseEsterelStatement(theparser);
67            } else {
68                throw new EsterelParserException("No Parser");
69            }
70        }
71        /**
72         * Returns a state representation of the esterel statement.
73         * @param anInitialState the esterel module which
74         * becomes a KIEL Statechart
75         * @param isFinalState is true if the new state has to be
76         * a final state
77         * @param theLocalEvents Stack with the local events
78         * @param theLocalVariables Stack with all local variables
79         * @param theTreeTrapSignals
80         * Stack with all tree trap signals.
81         * @throws Esterel2StudioException
82         * if the conversion to Kiel is not working. <br>
83         * Sideeffects: none
84         */
85        public final kiel.dataStructure.State convertToKiel(
86            final EsterelModule anEsterelModule,
87            final Arraylist theLocalEvents,
88            final Arraylist theLocalVariables,
89            final Arraylist theTreeTrapSignals)
90            throws Esterel2StudioException {
91        InitialArc aInitialArc = new InitialArc();
92    }
93}
94
95    /**
96     * Converts a State to an EsterelModule.
97     * @param anInitialState the initial state
98     * @param theTarget the target state
99     * @param theLabelOfTransition the label of a transition
100    Variable anVar = this.myVariable
101    .convertToKiel(anEsterelModule, theLocalEvents, theLocalVariables);
102
103    /**
104     * Parses a Document and set the class variables.
105     * cbrIt calls also the super constructor<code>EsterelStatement
106     * </code>.
107     * @param theparser theparser
108     */
109    public AssignEsterelStatement(ESTERELParser theparser) {
110        * Changes <code>EsterelParser</code> class variables
111        * @param theparser an intace of a EsterelParser
112        * Throws EsterelParserException if the parser is only delivered
113        * @see kiel.fileinterface.estrel.EsterelParserException
114        * @see EsterelStatement(EsterelParser)
115        * @see EsterelStatement(EsterelParser theparser)
116        * @see kiel.fileinterface.estrel.EsterelParser
117        */
118        public AssignEsterelStatement(final EsterelParser theparser)
119            throws EsterelParserException {
120            super(theparser);
121            this.myExpression = null;
122            this.myVariable = null;
123            if (theparser != null) {
124                this.parseEsterelStatement(theparser);
125            } else {
126                throw new Esterel2StudioException(
127                    this.getEsterelStatementName()
128                    + " "
129                    + " no expression or variable");
130            }
131        }
132        /**
133         * Implements the abstract methode from <code>EsterelStatement</code>.
134         * Parses an esterel statement in an <code>org.jdom.Document</code>
135         * representation of an .exp file
136         * <code> EsterelParser.createOR(
137         * this.getEsterelStatementName() + " state",
138         * new Node[] {anInitialState, theEndState});
139         */
140        public void setLabel(theLabelOfTransition) {
141            if (isFinalState) {
142                return StateChartHelpers.createFinalOR(
143                    this.getEsterelStatementName() + " state",
144                    new Node[] {anInitialState, theEndState});
145            } else {
146                return StateChartHelpers.createOR(
147                    this.getEsterelStatementName() + " state",
148                    new Node[] {anInitialState, theEndState});
149            }
150        }
151    }
152}

```


C.2. Paket kiel.fileInterface.estrel.estrel2estudio

C.2.3. AwaitEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.3 dargestellt.

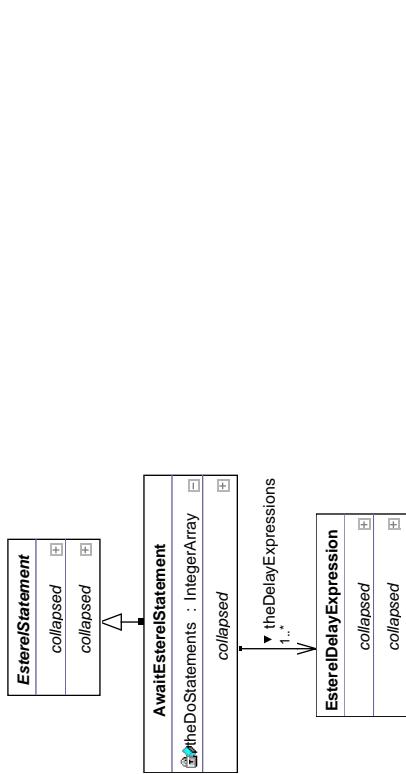


Abbildung C.3.: Klassendiagramm AwaitEsterelStatement

```

Aufistung C.8: Die Klasse AwaitEsterelStatement

package kiel.fileInterface.estrel.estrel12estudio;
// import org.jdom.Element;
import java.util.ArrayList;
// import kiel.datastructure.FinalSimpleState;
import kiel.datastructure.InitialC;
import kiel.datastructure.initState;
import kiel.datastructure.Node;
import kiel.datastructure.SimpleState;
import kiel.datastructure.State;
import kiel.datastructure.Transition;
import kiel.fileInterface.estrel.EsterelParser;
import kiel.fileInterface.estrel.EsterelParserException;
import kiel.util.DONRHelpers;
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.await s.await s.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>
 * $Revision: 1.55 $ last modified $Date: 2006/02/06 18:35:28 $
 * $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
 */

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

40   * *** empty log message ***
41   *
42   * Revision 1.9 2005/01/14 16:42:26 lku
43   * *** empty log message ***
44   *
45   * <br>
46   * Revision 1.8 2005/01/12 15:00:00 lku
47   *
48   * <br>
49   * add javadoc + check for project manual code conventions
50   *
51   * Revision 1.10
52   * change delayExpression from String to EsterelDelayExpression
53   * also major changes in set DelayExpression and convertToKiel
54   * add doStatement for await x do p
55   * add case
56   */
57   public class AwaitEsterelStatement extends EsterelStatement {
58       /**
59        * The value of the Expressions */
60       private EsterelDelayExpression[] delayExpression;
61       /**
62        * the do statements .
63       */
64       private int[] theDoStatement;
65       /**
66        * Changes <code>EsterelParser</code>>class variables
67       * @param theparser an instance of a EsterelParser
68       * @throws EsterelParserException is only delivered
69       * @see kiel.fileInterface.estrel.EsterelParserException
70       * @see #parseEsterelStatement(EsterelParser)
71       * @see EsterelStatement#EsterelStatement(EsterelParser)
72       * @see kiel.fileInterface.estrel.EsterelParser
73       */
74       public AwaitEsterelStatement(final EsterelParser theparser)
75           throws EsterelParserException {
76           super(theparser);
77           if (theparser != null) {
78               this.parseEsterelStatement(theparser);
79           } else {
80               throw new EsterelParserException("No Parser");
81           }
82       }
83       /**
84        * Returns a state representation of the estrel statement.
85        * @param anEsterelModule the estrel module which
86        * becomes a KIEL Statechart
87       */
88       public AwaitEsterelStatement(final EsterelParser theparser) {
89           /**
90            * Returns a state representation of the estrel statement.
91            * @param anEsterelModule the estrel module which
92            * becomes a KIEL Statechart
93           */
94       }
95   }

*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 theTreeTrapSignals
* @param theTrapSignals
* @throws Esterel2StudioException
* if the conversion to Kiel is not working. <br>
* Sideeffects: none
* @return a State
* @see kiel.fileInterface.estrel2studio.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<theTreeTrapSignals>
) throws Esterel2StudioException {
    InitialArc initialArc = new InitialArc();
    InitialState initialState = new InitialState();
    SimpleState theStartState = new SimpleState();
    State[] theEndStates = new State[theDelayExpression.length];
    for (int theCaseIndex = 0;
        theCaseIndex < this.delayExpression.length;
        theCaseIndex++) {
        if (this.theDoStatement[theCaseIndex] == -1) {
            theEndStates[theCaseIndex] = new FinalSimpleState();
        } else {
            theEndStates[theCaseIndex] =
                (EsterelStatement)
                    (anEsterelModule.getEsterelProgram().get(
                        this.theDoStatement[theCaseIndex]
                    ).convertToKiel(
                        this.theStartState,
                        theCaseIndex + 1,
                        theEndStates[theCaseIndex]));
        }
    }
    Transition awaitTransition =
        StatChartHelpers.createSA(
            theStartState,
            true,
            theLocalEvents,
            theLocalVariables,
            theTreeTrapSignals);
    awaitTransition.setLabel(
        this.delayExpression[theCaseIndex].convertToKiel(
            anEsterelModule,
            theStartState,
            theCaseIndex + 1,
            theEndStates[theCaseIndex]));
}

*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 theTreeTrapSignals
* @param theTrapSignals
* @throws Esterel2StudioException
* if the conversion to Kiel is not working. <br>
* Sideeffects: none
* @return a State
* @see kiel.fileInterface.estrel2studio.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<theTreeTrapSignals>
) throws Esterel2StudioException {
    InitialArc initialArc = new InitialArc();
    InitialState initialState = new InitialState();
    SimpleState theStartState = new SimpleState();
    State[] theEndStates = new State[theDelayExpression.length];
    for (int theCaseIndex = 0;
        theCaseIndex < this.delayExpression.length;
        theCaseIndex++) {
        if (this.theDoStatement[theCaseIndex] == -1) {
            theEndStates[theCaseIndex] = new FinalSimpleState();
        } else {
            theEndStates[theCaseIndex] =
                (EsterelStatement)
                    (anEsterelModule.getEsterelProgram().get(
                        this.theDoStatement[theCaseIndex]
                    ).convertToKiel(
                        this.theStartState,
                        theCaseIndex + 1,
                        theEndStates[theCaseIndex]));
        }
    }
    Transition awaitTransition =
        StatChartHelpers.createSA(
            theStartState,
            true,
            theLocalEvents,
            theLocalVariables,
            theTreeTrapSignals);
    awaitTransition.setLabel(
        this.delayExpression[theCaseIndex].convertToKiel(
            anEsterelModule,
            theStartState,
            theCaseIndex + 1,
            theEndStates[theCaseIndex]));
}

```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

        awaitTransition,
        awaitExpression(theCaseIndex).convertToKiel(
            anEsterelModule,
            theLocalEvents) */
```

1 }
 initialArc.setSource(anInitialState);
 initialArc.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);
}

160 }
 /**
 * Implements the abstract methode from <code>EsterelStatement</code>
 * Parses an esterel statement in an <code>jdom.Document</code>
 * representation of an .exp file
 * if you have not called the constructor
 * <code> EsterelStatement EsterelParser theparser</code>
 * use the methode <code>parseEsterelStatement</code>
 * before <code>parseEsterelParser</code>
 *
<code>Effects:
 * 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 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.aDelayExpression =
 new EsterelDelayExpression(fanElementList.size());
 this.theBottomElement = new int[fanElementList.size()];
 // set default values
 for (int i = 0; i < anElementList.size(); i++) {
 this.theDelayExpression[i] = null;
 this.theStatement[i] = -1;
 }
 }
}

C. Java Code für die Transformation von Esterel in SyncCharts

```

 * @see kiel.fileInterface.estrel.EsterelParser
 */
public final void setSubStatements(final EsterelParser theparser
throws EsterelParserException {
    List<XMLElement> anElementList =
        DOMHelpers.getELEMents(
            theparser.getXMLDocument());
    270   this.getxpathSearchString();
    for (int theCaseIndex = 0;
        theCaseIndex < anElementList.size();
        theCaseIndex++) {
        if ((Element) anElementList.get(theCaseIndex)).getName()
            == "PredicatedStatement") {
            String aPredicatedStatementXPathSearchString =
                this.getxpathSearchString(",
                + "[Local-name()='",
                + "'] [@id='",
                + (
                    (Element) anElementList.get(
                        theCaseIndex)).getAttributeValue(
                            "id")
                + "']");
            280   List<XMLElement> aPredicatedElementList =
                DOMHelpers.getELEMents(
                    theparser.getXMLDocument(),
                    aPredicatedStatementXPathSearchString);
            if (aPredicatedElementList.size() > 1) {
                this.theDoStatement[theCaseIndex] = theparser.getAddAt();
                theparser.getEsterelModule(
                    theparser.getActualEsterelModule())
                    .addStatementToModule(
                        aPredicatedElementList.size() - 1);
                theparser.createStatement(
                    (Element) aPredicatedElementList.get(0)));
            } else {
                290   this.theDoStatement[theCaseIndex] = -1;
            }
        }
    }
}
300
} //public class AwaitEsterelStatement

```

C.2.4. DoUpToEstrelStatement

Der Klassenaufbau ist in der Abbildung C.2.4 dargestellt.

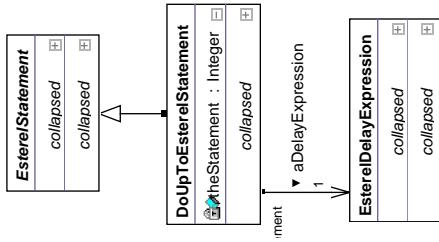


Abbildung C.4.: Klassendiagramm DoUpToEstrelStatement

Auflistung C.9: Die Klasse DoUpToEstrelStatement

```

package kiel.fileInterface.estrel.estrel2estudio;
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.estrel.EstrelParser;
import kiel.fileInterface.estrel.EstrelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StatechartHelpers;
/
import org.json.Element;

/*
 * <p>Implements the estrel <code>DoUpTo</code> statement.</p>
 * <p>Copyright (c) 2005</p>
 * <p>Company: Uni Kiel</p>
 * <p>Author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuhl</a>
 * Version $Revision: 1.35 $ last modified $Date: 2006/02/06 18:35:28 $
 * <br>
 */
public class DoUpToEstrelStatement extends EstrelStatement {
    /**
     * The value of the Expression.*/
    private EstrelDelayExpression delayExpression;
    /**
     * the statement which to DoUpTo.
     */
    private int theStatement;
    /**
     * the standard constructor.
     */
}

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

        State theDoupToDstate =
        (
            (EsterelStatement)
            (
                anEsterelModule.getEsterelProgram().get(
                    "this.theStatement").convertTokiel(
                    anEsterelModule,
                    false,
                    theLocalEvents,
                    theLocalVariables,
                    theTreesTrapSignals,
                    Transition.asTransition =
                        StateChartHelpers.createSA(theDoupToDstate, 1, theEndState);
                    aSATransition.setLabel(
                        this.delayExpression.convertTokiel(
                            anEsterelModule,
                            theLocalEvents,
                            theLocalVariables,
                            null));
                    alInitialArc.setSource(anInitialState);
                    alInitialArc.setTarget(theDoupToDstate);
                    // if theDoupToDstate is not DoupToD
                    // then a normal termination with lowest priority is used
                    if (isFinalState) {
                        return StateChartHelpers.createFinalOR(
                            this.getEsterelStatementName() + "state",
                            new Node[] {anInitialState, theEndState, theDoupToDstate});
                    }
                    return StateChartHelpers.createOR(
                        this.getEsterelStatementName() + "state",
                        new Node[] {anInitialState, theEndState, theDoupToDstate});
                }
            )
        );
    }

    /**
     * Parses a Document and set the class variables.
     * <br>It calls also the super constructor</code>EsterelStatement
     * (EsterelParser theparser)</code> .
     * <br>Calls the <code>parseEsterelStatement</code> methode
     * <br>Sideeffects:
     * Changes <code>EsterelParser</code>class variables
     * @param theparser an instance of a EsterelParser
     * @throws EsterelParserException is only delivered
     * @see kiel.fileInterface.estrel.EsterelParserException
     * @see #parseEsterelStatement(EsterelParser)
     * @see EsterelStatement#EsterelStatement(EsterelParser)
     * @see kiel.fileInterface.estrel.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 estrel statement.
     * @param anEsterelModule the estrel 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 Esterel2StudioException
     * if the conversion to Kiel is not working. <br>
     * Sideeffects: none
     * @return a State
     * @see kiel.fileInterface.estrel.estrel2studio.EsterelModule
     */
    public final kiel.datastructure.State convertTokiel(
        final EsterelModule anEsterelModule,
        final boolean isFinalState,
        final ArrayList<theLocalEvents>,
        final ArrayList<theLocalVariables>,
        final ArrayList<theTreesTrapSignals>)
        throws Esterel2StudioException {
        InitialArc alInitialArc = new InitialArc();
        InitialState anInitialState = new InitialState();
        FinalSimpleState theEndState = new FinalSimpleState();
        List<anElementList> =

```

156

```

DOMHelpers.getElements(
    theparser.getXMLDocument(),
    this.getXPathSearchString());
    // the statement which may be Dolptoed is before the last element
    if (anElementList.size() == 2) {
        this.theStatement = theparser.getAddit();
        theparser.getEstrelModule(
            theparser.getActualEstrelModule()
                .addStatementToModule(
                    theparser.getAddit(),
                    theparser.createStatement(((Element) (anElementList.get(0))))));
    }
}

150
    /**
     * Implements the abstract methode from <code>EstrelStatement</code>.
     * Returns a string representation of an estrel statement.
     * @param anEstrelModule an EstrelModule an estrel modul representation
     * @return a String representation of an estrel statement
     * @see kiel.fileInterface.estrel.estrel2estudio.EstrelStatement
     */
    public final String toString(final EstrelModule anEstrelModule) {
        String result =
            "do \n"
            + (
                (EstrelStatement)
                    (anEstrelModule.getEstrelProgram().get(
                        this.theStatement)).toString(
                            anEstrelModule));
        result += "\n upto " + this.delayExpression.toString(anEstrelModule);
        result += "\n end ";
        return result;
    }
}

155
    /**
     * Fills the statements in the estrel module.
     * In this class it is fills the do Statements if there are any
     * <br> Sideeffects: none
     * @param theparser the actual parser
     * @throws EstrelParserException is only delivered
     * @see kiel.fileInterface.estrel.EstrelParserException
     */
    public final void setSubStatements(final EstrelParser theparser)
        throws EstrelParserException {
        List anElementList =
            DOMHelpers.getElements(
                theparser.getXMLDocument(),

```

C.2.5. DoWatchingEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.5 dargestellt.

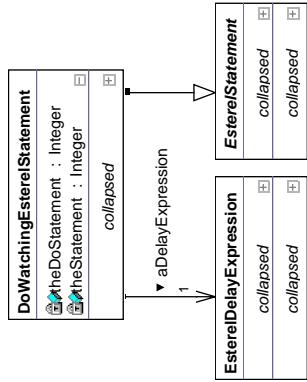


Abbildung C.5.: Klassendiagramm DoWatchingEsterelStatement

Auflistung C.10: Die Klasse DoWatchingEsterelStatement

```

/*
 * package kiel.fileinterface.estrel.estrel2studio;
 * 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.estrel.EsterelParser;
import kiel.fileInterface.estrel.EsterelParserException;
import kiel.util.DOHHelpers;
import kiel.util.StateChartHelpers;
import org.json.Element;
*/
/*>Implements the estrel <code>DoWatching</code> statement.</p>
 * <p>Copyright: Copyright (c) 2005</p>
 * <p>Company: Uni Kiel</p>
 * <br>Author <a href="mailto:luke@informatik.uni-kiel.de">Lars Kuehl</a>
 * <br>Version $Revision: 1.35 $ last modified $Date: 2006/02/06 18:35:28 $
 * <br>Sideeffects:
 */
public class DoWatchingEsterelStatement extends EsterelStatement {
    /**
     * The value of the Expression.*/
    private EsterelDelayExpression aDelayExpression;
    /**
     * the do statement.
     */
    private int theDoStatement;
    /**
     * the statement which to DoWatching.
     */
    private int theStatement;
    /**
     * private int theStatement;
     */
    /**
     * the standard constructor.
     */
    public DoWatchingEsterelStatement() {
        super();
    }
    /**
     * Parses a Document and set the class variables.
     */
    public void parseEsterelStatement() {
        //Parse the Document
        //Call the super constructor
        //Parse the expression
        //Call the DoWatching method
    }
}

```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

C. Java Code für die Transformation von Esterel in SyncCharts

```

* 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.estrel.EsterelParserException
*         * @see EsterelStatement#parseEsterelStatement(EsterelParser)
*         * @see EsterelStatement#parseEsterelStatement(EsterelParser)
*         * @see kiel.fileInterface.estrel.EsterelParser
*         * @see org.json.Document
*/
170    public final void parseEsterelStatement(final EsterelParser theparser)
        throws EsterelParserException {
        List anElementList =
            DOMHelpers.getElements(
                theparser.getXMLDocument(),
                this.getXMLSearchString());
        if (anElementList.size() > 1) {
            //the statement to Dolatching
            theparser.inStatementCounter();
            // the delay expression
            this.delayExpression =
                new EsterelDelayExpression(
                    theparser,
                    "/*[@id=" +
                    ((Element) anElementList.get(1)).getAttributeValue(
                        "id") +
                    "+ "]*/",
                    theparser.getXMLDocument());
            // timeout statement
            if (anElementList.size() > 2) {
                theparser.inStatementCounter();
            }
        }
    }

    /**
     * Fills the statements in the estrel 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.estrel.EsterelParserException
     */
    public final void setSubStatements(final EsterelParser theparser
        throws EsterelParserException {
        List anElementList =
            DOMHelpers.getElements(
                theparser.getXMLDocument(),
                this.getXMLSearchString());
        if (anElementList.size() > 1) {
            this.get XPathSearchString();
            // the statement which may be Dolatching is before the last element
            if (anElementList.size() > 1) {
                this.theStatement = theparser.getAddAt();
                theparser.getEsterelModule(
                    theparser.getActiveEsterelModule()
                        .addStatementToModule(
                            theparser.getAddAt(),
                            theparser.createStatement((Element) (anElementList.get(0))));
                )
            }
            // the do statement
            if (anElementList.size() > 2) {
                this.theDoStatement = theparser.getAddAt();
                theparser.getEsterelModule(
                    theparser.getActiveEsterelModule()
                        .addStatementToModule(
                            theparser.getAddAt(),
                            theparser.createStatement((Element) (anElementList.get(2))));
                )
            }
            /**
             * Implements the abstract method from <code>EsterelStatement</code>.
             * Returns a string representation of an estrel statement.
             * @param anEsterelModule an estrel module representation
             * @return a String representation of an estrel statement
             * @see kiel.fileInterface.estrel.EsterelStatement
             * @see kiel.fileInterface.estrel.EsterelModule
             */
            public final String toString(final EsterelModule anEsterelModule) {
                String result =
                    "do \n"
                    + (
                        (EsterelStatement)
                            anEsterelModule.getEsterelProgram().get(
                                this.theStatement)).toString(
                                anEsterelModule);
                result += "\n in watching "
                + this.delayExpression.toString(anEsterelModule);
                if (this.theDoStatement != -1) {
                    result += " timeout\n"
                    + (
                        (EsterelStatement) anEsterelModule.getEsterelProgram().get(
                            this.theDoStatement)).toString(
                                anEsterelModule);
                }
                result += "\n end ";
                return result;
            }
        }
    }

    /**
     * @param theparser theparser
     */
    public final void parseEsterelStatement(final EsterelParser theparser
        180
        190
        200
        210
        220
        230
        240
        250
        260
        270
        280
        290
        300
        310
        320
        330
        340
        350
        360
        370
        380
        390
        400
        410
        420
        430
        440
        450
        460
        470
        480
        490
        500
        510
        520
        530
        540
        550
        560
        570
        580
        590
        600
        610
        620
        630
        640
        650
        660
        670
        680
        690
        700
        710
        720
        730
        740
        750
        760
        770
        780
        790
        800
        810
        820
        830
        840
        850
        860
        870
        880
        890
        900
        910
        920
        930
        940
        950
        960
        970
        980
        990
        1000
        1010
        1020
        1030
        1040
        1050
        1060
        1070
        1080
        1090
        1100
        1110
        1120
        1130
        1140
        1150
        1160
        1170
        1180
        1190
        1200
        1210
        1220
        1230
        1240
        1250
        1260
        1270
        1280
        1290
        1300
        1310
        1320
        1330
        1340
        1350
        1360
        1370
        1380
        1390
        1400
        1410
        1420
        1430
        1440
        1450
        1460
        1470
        1480
        1490
        1500
        1510
        1520
        1530
        1540
        1550
        1560
        1570
        1580
        1590
        1600
        1610
        1620
        1630
        1640
        1650
        1660
        1670
        1680
        1690
        1700
        1710
        1720
        1730
        1740
        1750
        1760
        1770
        1780
        1790
        1800
        1810
        1820
        1830
        1840
        1850
        1860
        1870
        1880
        1890
        1900
        1910
        1920
        1930
        1940
        1950
        1960
        1970
        1980
        1990
        2000
        2010
        2020
        2030
        2040
        2050
        2060
        2070
        2080
        2090
        2100
        2110
        2120
        2130
        2140
        2150
        2160
        2170
        2180
        2190
        2200
        2210
        2220
        2230
        2240
        2250
        2260
        2270
        2280
        2290
        2300
        2310
        2320
        2330
        2340
        2350
        2360
        2370
        2380
        2390
        2400
        2410
        2420
        2430
        2440
        2450
        2460
        2470
        2480
        2490
        2500
        2510
        2520
        2530
        2540
        2550
        2560
        2570
        2580
        2590
        2600
        2610
        2620
        2630
        2640
        2650
        2660
        2670
        2680
        2690
        2700
        2710
        2720
        2730
        2740
        2750
        2760
        2770
        2780
        2790
        2800
        2810
        2820
        2830
        2840
        2850
        2860
        2870
        2880
        2890
        2900
        2910
        2920
        2930
        2940
        2950
        2960
        2970
        2980
        2990
        3000
        3010
        3020
        3030
        3040
        3050
        3060
        3070
        3080
        3090
        3100
        3110
        3120
        3130
        3140
        3150
        3160
        3170
        3180
        3190
        3200
        3210
        3220
        3230
        3240
        3250
        3260
        3270
        3280
        3290
        3300
        3310
        3320
        3330
        3340
        3350
        3360
        3370
        3380
        3390
        3400
        3410
        3420
        3430
        3440
        3450
        3460
        3470
        3480
        3490
        3500
        3510
        3520
        3530
        3540
        3550
        3560
        3570
        3580
        3590
        3600
        3610
        3620
        3630
        3640
        3650
        3660
        3670
        3680
        3690
        3700
        3710
        3720
        3730
        3740
        3750
        3760
        3770
        3780
        3790
        3800
        3810
        3820
        3830
        3840
        3850
        3860
        3870
        3880
        3890
        3900
        3910
        3920
        3930
        3940
        3950
        3960
        3970
        3980
        3990
        4000
        4010
        4020
        4030
        4040
        4050
        4060
        4070
        4080
        4090
        4100
        4110
        4120
        4130
        4140
        4150
        4160
        4170
        4180
        4190
        4200
        4210
        4220
        4230
        4240
        4250
        4260
        4270
        4280
        4290
        4300
        4310
        4320
        4330
        4340
        4350
        4360
        4370
        4380
        4390
        4400
        4410
        4420
        4430
        4440
        4450
        4460
        4470
        4480
        4490
        4500
        4510
        4520
        4530
        4540
        4550
        4560
        4570
        4580
        4590
        4600
        4610
        4620
        4630
        4640
        4650
        4660
        4670
        4680
        4690
        4700
        4710
        4720
        4730
        4740
        4750
        4760
        4770
        4780
        4790
        4800
        4810
        4820
        4830
        4840
        4850
        4860
        4870
        4880
        4890
        4900
        4910
        4920
        4930
        4940
        4950
        4960
        4970
        4980
        4990
        5000
        5010
        5020
        5030
        5040
        5050
        5060
        5070
        5080
        5090
        5100
        5110
        5120
        5130
        5140
        5150
        5160
        5170
        5180
        5190
        5200
        5210
        5220
        5230
        5240
        5250
        5260
        5270
        5280
        5290
        5300
        5310
        5320
        5330
        5340
        5350
        5360
        5370
        5380
        5390
        5400
        5410
        5420
        5430
        5440
        5450
        5460
        5470
        5480
        5490
        5500
        5510
        5520
        5530
        5540
        5550
        5560
        5570
        5580
        5590
        5600
        5610
        5620
        5630
        5640
        5650
        5660
        5670
        5680
        5690
        5700
        5710
        5720
        5730
        5740
        5750
        5760
        5770
        5780
        5790
        5800
        5810
        5820
        5830
        5840
        5850
        5860
        5870
        5880
        5890
        5900
        5910
        5920
        5930
        5940
        5950
        5960
        5970
        5980
        5990
        6000
        6010
        6020
        6030
        6040
        6050
        6060
        6070
        6080
        6090
        6100
        6110
        6120
        6130
        6140
        6150
        6160
        6170
        6180
        6190
        6200
        6210
        6220
        6230
        6240
        6250
        6260
        6270
        6280
        6290
        6300
        6310
        6320
        6330
        6340
        6350
        6360
        6370
        6380
        6390
        6400
        6410
        6420
        6430
        6440
        6450
        6460
        6470
        6480
        6490
        6500
        6510
        6520
        6530
        6540
        6550
        6560
        6570
        6580
        6590
        6600
        6610
        6620
        6630
        6640
        6650
        6660
        6670
        6680
        6690
        6700
        6710
        6720
        6730
        6740
        6750
        6760
        6770
        6780
        6790
        6800
        6810
        6820
        6830
        6840
        6850
        6860
        6870
        6880
        6890
        6900
        6910
        6920
        6930
        6940
        6950
        6960
        6970
        6980
        6990
        7000
        7010
        7020
        7030
        7040
        7050
        7060
        7070
        7080
        7090
        7100
        7110
        7120
        7130
        7140
        7150
        7160
        7170
        7180
        7190
        7200
        7210
        7220
        7230
        7240
        7250
        7260
        7270
        7280
        7290
        7300
        7310
        7320
        7330
        7340
        7350
        7360
        7370
        7380
        7390
        7400
        7410
        7420
        7430
        7440
        7450
        7460
        7470
        7480
        7490
        7500
        7510
        7520
        7530
        7540
        7550
        7560
        7570
        7580
        7590
        7600
        7610
        7620
        7630
        7640
        7650
        7660
        7670
        7680
        7690
        7700
        7710
        7720
        7730
        7740
        7750
        7760
        7770
        7780
        7790
        7800
        7810
        7820
        7830
        7840
        7850
        7860
        7870
        7880
        7890
        7900
        7910
        7920
        7930
        7940
        7950
        7960
        7970
        7980
        7990
        8000
        8010
        8020
        8030
        8040
        8050
        8060
        8070
        8080
        8090
        8100
        8110
        8120
        8130
        8140
        8150
        8160
        8170
        8180
        8190
        8200
        8210
        8220
        8230
        8240
        8250
        8260
        8270
        8280
        8290
        8300
        8310
        8320
        8330
        8340
        8350
        8360
        8370
        8380
        8390
        8400
        8410
        8420
        8430
        8440
        8450
        8460
        8470
        8480
        8490
        8500
        8510
        8520
        8530
        8540
        8550
        8560
        8570
        8580
        8590
        8600
        8610
        8620
        8630
        8640
        8650
        8660
        8670
        8680
        8690
        8700
        8710
        8720
        8730
        8740
        8750
        8760
        8770
        8780
        8790
        8800
        8810
        8820
        8830
        8840
        8850
        8860
        8870
        8880
        8890
        8900
        8910
        8920
        8930
        8940
        8950
        8960
        8970
        8980
        8990
        9000
        9010
        9020
        9030
        9040
        9050
        9060
        9070
        9080
        9090
        9100
        9110
        9120
        9130
        9140
        9150
        9160
        9170
        9180
        9190
        9200
        9210
        9220
        9230
        9240
        9250
        9260
        9270
        9280
        9290
        9300
        9310
        9320
        9330
        9340
        9350
        9360
        9370
        9380
        9390
        9400
        9410
        9420
        9430
        9440
        9450
        9460
        9470
        9480
        9490
        9500
        9510
        9520
        9530
        9540
        9550
        9560
        9570
        9580
        9590
        9600
        9610
        9620
        9630
        9640
        9650
        9660
        9670
        9680
        9690
        9700
        9710
        9720
        9730
        9740
        9750
        9760
        9770
        9780
        9790
        9800
        9810
        9820
        9830
        9840
        9850
        9860
        9870
        9880
        9890
        9900
        9910
        9920
        9930
        9940
        9950
        9960
        9970
        9980
        9990
        10000
        10010
        10020
        10030
        10040
        10050
        10060
        10070
        10080
        10090
        10100
        10110
        10120
        10130
        10140
        10150
        10160
        10170
        10180
        10190
        10200
        10210
        10220
        10230
        10240
        10250
        10260
        10270
        10280
        10290
        10300
        10310
        10320
        10330
        10340
        10350
        10360
        10370
        10380
        10390
        10400
        10410
        10420
        10430
        10440
        10450
        10460
        10470
        10480
        10490
        10500
        10510
        10520
        10530
        10540
        10550
        10560
        10570
        10580
        10590
        10600
        10610
        10620
        10630
        10640
        10650
        10660
        10670
        10680
        10690
        10700
        10710
        10720
        10730
        10740
        10750
        10760
        10770
        10780
        10790
        10800
        10810
        10820
        10830
        10840
        10850
        10860
        10870
        10880
        10890
        10900
        10910
        10920
        10930
        10940
        10950
        10960
        10970
        10980
        10990
        11000
        11010
        11020
        11030
        11040
        11050
        11060
        11070
        11080
        11090
        11100
        11110
        11120
        11130
        11140
        11150
        11160
        11170
        11180
        11190
        11200
        11210
        11220
        11230
        11240
        11250
        11260
        11270
        11280
        11290
        11300
        11310
        11320
        11330
        11340
        11350
        11360
        11370
        11380
        11390
        11400
        11410
        11420
        11430
        11440
        11450
        11460
        11470
        11480
        11490
        11500
        11510
        11520
        11530
        11540
        11550
        11560
        11570
        11580
        11590
        11600
        11610
        11620
        11630
        11640
        11650
        11660
        11670
        11680
        11690
        11700
        11710
        11720
        11730
        11740
        11750
        11760
        11770
        11780
        11790
        11800
        11810
        11820
        11830
        11840
        11850
        11860
        11870
        11880
        11890
        11900
        11910
        11920
        11930
        11940
        11950
        11960
        11970
        11980
        11990
        12000
        12010
        12020
        12030
        12040
        12050
        12060
        12070
        12080
        12090
        12100
        12110
        12120
        12130
        12140
        12150
        12160
        12170
        12180
        12190
        12200
        12210
        12220
        12230
        12240
        12250
        12260
        12270
        12280
        12290
        12300
        12310
        12320
        12330
        12340
        12350
        12360
        12370
        12380
        12390
        12400
        12410
        12420
        12430
        12440
        12450
        12460
        12470
        12480
        12490
        12500
        12510
        12520
        12530
        12540
        12550
        12560
        12570
        12580
        12590
        12600
        12610
        12620
        12630
        12640
        12650
        12660
        12670
        12680
        12690
        12700
        12710
        12720
        12730
        12740
        12750
        12760
        12770
        12780
        12790
        12800
        12810
        12820
        12830
        12840
        12850
        12860
        12870
        12880
        12890
        12900
        12910
        12920
        12930
        12940
        12950
        12960
        12970
        12980
        12990
        13000
        13010
        13020
        13030
        13040
        13050
        13060
        13070
        13080
        13090
        13100
        13110
        13120
        13130
        13140
        13150
        13160
        13170
        13180
        13190
        13200
        13210
        13220
        13230
        13240
        13250
        13260
        13270
        13280
        13290
        13300
        13310
        13320
        13330
        13340
        13350
        13360
        13370
        13380
        13390
        13400
        13410
        13420
        13430
        13440
        13450
        13460
        13470
        13480
        13490
        13500
        13510
        13520
        13530
        13540
        13550
        13560
        13570
        13580
        13590
        13600
        13610
        13620
        13630
        13640
        13650
        13660
        13670
        13680
        13690
        13700
        13710
        13720
        13730
        13740
        13750
        13760
        13770
        13780
        13790
        13800
        13810
        13820
        13830
        13840
        13850
        13860
        13870
        13880
        13890
        13900
        13910
        13920
        13930
        13940
        13950
        13960
        13970
        13980
        13990
        14000
        14010
        14020
        14030
        14040
        14050
        14060
        14070
        14080
        14090
        14100
        14110
        14120
        14130
        14140
        14150
        1
```

C.2.6. EmitEstrelStatement

Der Klassenaufbau ist in der Abbildung C.2.6 dargestellt.

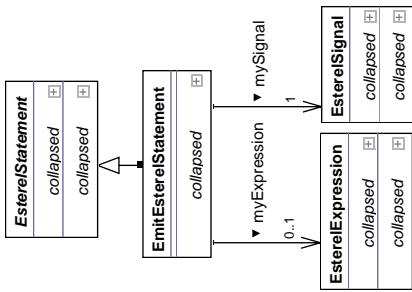


Abbildung C.6.: Klassendiagramm EmitEstrelStatement

```

Auflistung C.11: Die Klasse EmitEstrelStatement

package kiel.fileInterface.estrel.estrel2estudio;
import org.jdom.Element;
20
/*
 */
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.dataStructure.estrel.EsterelParser;
import kiel.fileInterface.estrel.EsterelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StatechartHelpers;
10
public class EmitEstrelStatement {
    /*
     * This is a subclass of <code>EstrelStatement</code>
     * classes implements the emit statement.
     */
    /*
     * Copyright: Copyright (c) 2004
     * Company: Uni Kiel
     */
    /*
     * Author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl </a>
     * $Date: 2006/02/13 14:00:04 $
     */
    /*
     * Overview Revision: 1.58 $ last modified $Date: 2006/02/13 14:00:04 $
     */
    /*
     * The expression of the signal to emit. */
}
  
```

C. Java Code für die Transformation von Esterel in SyncCharts

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

C.2.7. Esterel2EstudioException

Auflistung C.12: Die Klasse Esterel2EstudioException

```

package kiel.fileInterface.estrel.estrel2estudio;
import javax.swing.JOptionPane;

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

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.estrel.estrel2estudio;

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

/**
 * <p>
 * Used to load and store user definable properties.
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 * <p>
 * Author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kümmel</a>
 * @version $Revision: 1.11 $
 * /
public final class Esterel2StudioProperties {

    /**
     * The default resource 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 SEPARATOR = "ArgumentSeparator";
    /**
     * 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");
    }
    /**
     * @return true, if option is set to true.
     */
    public static boolean getProcedures() {
        return properties.getProperty("Procedures");
    }
    /**
     * @return true, if option is set to true.
     */
    public static boolean getTasks() {
        return properties.getProperty("Tasks");
    }
    /**
     * @return true, if option is set to true.
     */
    public static final String PROPERTYFILE = System.getProperty("user.home") +
        + File.separator +
        + ".kiel";
    /**
     * The post string for new signals
     */
    public static String getPostString() {
        return properties.getProperty("NewSignalsPostString");
    }
    /**
     * @return the separator in calls
     */
}

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

/*
 public static String getSeparatorString() {
    return properties.getProperty(SEPARATOR);
 }
 /**
 * @return if automatic optimization is on means true.
 */
 110  /*
 public static String getPreString() {
    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
 */
 120  /*
 public static boolean load() {
    boolean success;
    Properties properties = new Properties();
    properties.loadDefaults();
    if (!loadProperties()) {
        success = copyDefaults();
    } else {
        success = true;
    }
    return success;
 }

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

 /**
 * Copies properties to user specific file.
 */
 140  public static void copyDefaults() {
    InputStream is;
    try {
        is = Esterel2EstudioProperties
            .getResourcesAsStream(DEFAULTRESOURCE);
        boolean success = true;
        try {
            File f;
            try {
                f = new FileWriter(
                    PROPERTIESFILE);
                int c = is.read();
                while (c >= 0) {
                    fw.write(c);
                    c = is.read();
                }
                fw.flush();
                is.close();
                fw.close();
            } catch (IOException storeException) {
                success = false;
            }
        } catch (IOException e) {
            success = false;
        }
    } catch (IOException e) {
        success = false;
    }
    return success;
}

 /**
 * Loads the defaults from resource file.
 */
 160  private static Properties getDefaults() {
    Properties defaults = new Properties();
    try {
        defaults
            .load(Esterel2EstudioProperties
                .getClass()
                .getResourcesAsStream(DEFAULTRESOURCE));
    } catch (IOException e) {
        defaults.clear();
    }
    return defaults;
}

 /**
 * Loads properties from user specific file.
 */
 170  private static void loadProperties() {
    try {
        properties.load(new FileInputStream(
            PROPERTYFILE));
    } catch (IOException loadException) {
        success = false;
    }
    return;
}

 /**
 * Loads properties from user specific file.
 */
 180  private static void loadProperties() {
    boolean success = true;
    try {
        properties.load(new FileInputStream(
            PROPERTYFILE));
    } catch (IOException loadException) {
        success = false;
    }
    return;
}

 /**
 * Copies properties to user specific file.
 */
 190  private static void copyProperties() {
    try {
        is = Esterel2EstudioProperties
            .getResourcesAsStream(DEFAULTRESOURCE);
        boolean success = true;
        try {
            File f;
            try {
                f = new FileWriter(
                    PROPERTIESFILE);
                int c = is.read();
                while (c >= 0) {
                    fw.write(c);
                    c = is.read();
                }
                fw.flush();
                is.close();
                fw.close();
            } catch (IOException storeException) {
                success = false;
            }
        } catch (IOException e) {
            success = false;
        }
    } catch (IOException e) {
        success = false;
    }
    return;
}

 /**
 * This is the constructor.
 */
 200  private Esterel2EstudioProperties() {
}

```

C.2.9. EstrelConstant

Der Klassenaufbau ist in der Abbildung C.2.9 dargestellt.

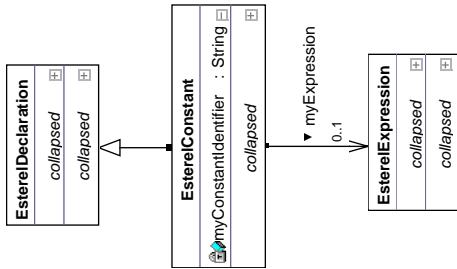


Abbildung C.7.: Klassediagramm EstrelConstant

Auflistung C.14: Die Klasse EstrelConstant

```

package kiel.fileInterface.estrel.estrel2estudio;
/*
 * <p> It represents estrel constants .</p>
 * <p>Copyright: Copyright (c) 2005</p>
 * <p>Company: Uni Kiel</p>
 * <p>Author <a href="mailto:lkw@informatik.uni-kiel.de">Lars Kuehl</a>
 * Version $Revision: 1.35 $ last modified $Date: 2006/02/06 18:35:29 $
 */
public class EstrelConstant
extends EsterelDeclaration {
    /**
     * The type of the constant. (optional).
     */
    private EstrelTypeDeclaration myChannelType; // child1
    /**
     *The name of the constant.
     */
}
  
```

C. Java Code für die Transformation von Esterel in SyncCharts

```

private String myConstantIdentifier; // child 0
/**
 * The expression of the constant. (Optional).
 */
private EsterelExpression myExpression; // child 2
/** the standard constructor.
 */
40   /**
    * Creates an EsterelVariable and fills the class variables.
    * <br>Sideffects: 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
 * @param theLocalEvents
 * @param theLocalVariables
 * @param theLocalEvents which becomes a KIEL Statechart
70   /**
    * Stack with the local events
    * Stack with all local variables
    * @throws Esterel2StudioException
    * If the conversion to Kiel is not working. <br>
    * Sideffects: none
    * @return a State
    * @see kiel.fileInterface.esterel.esterel2studio.EsterelModule
    * @see kiel.dataStructure.State
    */
    public final Variable convertToKiel(
        final EsterelModule anEsterelModule,
        final ArrayList<Variable> theLocalEvents,
        final ArrayList<Variable> theLocalVariables)
        throws Esterel2StudioException {
    Variable result = null;
    if (result == null) {
        if (this.myChannelType != null) {
            if (this.myChannelType
                .get TypeName() == 0) {
                if (this.myExpression != null) {
                    result = new StringVariable(
                        this.myConstantIdentifier);
                }
            } else if (this.myChannelType
                .get TypeName() == 1) {
                result = new IntegerVariable(
                    this.myConstantIdentifier);
            }
        } else if (this.myChannelType
            .get TypeName() == 2) {
            if (this.myExpression != null) {
                result = new IntegerVariable(
                    this.myConstantIdentifier,
                    this.myExpression.convertToKiel(anEsterelModule),
                    theLocalEvents,
                    theLocalVariables);
            }
        } else if (this.myChannelType
            .get TypeName() == 3) {
            if (this.myExpression != null) {
                result = new IntegerVariable(
                    this.myConstantIdentifier,
                    this.myExpression.convertToKiel(anEsterelModule),
                    theLocalEvents,
                    theLocalVariables);
            }
        } else if (this.myChannelType
            .get TypeName() == 4) {
            if (this.myExpression != null) {
                result = new BooleanVariable(
                    this.myConstantIdentifier);
            }
        } else if (this.myChannelType
            .get TypeName() == 5) {
            if (this.myExpression != null) {
                result = new BooleanVariable(
                    this.myConstantIdentifier,
                    this.myExpression.convertToKiel(anEsterelModule),
                    theLocalEvents,
                    theLocalVariables);
            }
        } else if (this.myChannelType
            .get TypeName() == 6) {
            if (this.myExpression != null) {
                result = new FloatVariable(
                    this.myConstantIdentifier);
            }
        } else if (this.myChannelType
            .get TypeName() == 7) {
            if (this.myExpression != null) {
                result = new FloatVariable(
                    this.myConstantIdentifier,
                    this.myExpression.convertToKiel(anEsterelModule));
            }
        } else {
            result = new FloatVariable(
                this.myConstantIdentifier);
        }
    }
}
80   /**
    * After full support for Float this.myExpression
    * .convertToKiel(anEsterelModule);
    */
    public final Variable convertToKiel(
        final EsterelModule anEsterelModule,
        final ArrayList<Variable> theLocalEvents,
        final ArrayList<Variable> theLocalVariables)
        throws Esterel2StudioException {
    Variable result = null;
    if (result == null) {
        if (this.myChannelType != null) {
            if (this.myChannelType
                .get TypeName() == 0) {
                if (this.myExpression != null) {

```

```

result = new DoubleVariable(
    this.myConstantIdentifier);
/* After full support for double:
 * *this.myExpression
 * *convertToKiel(anEstrelModule);
 */
150     } else {
        result = new DoubleVariable(
            this.myConstantIdentifier);
    }
} else {
    // set the type to string
    if (this.myExpression != null) {
        result = new StringVariable(
            this.myConstantIdentifier,
            this.myExpression
                .convertToKielString(anEstrelModule));
    }
    /* throw new Estrel2EstudioException(
     * this.getFname() + "Unknown type : " +
     * this.myChannelType);
 */
160     } else {
        result = new StringVariable(
            this.myConstantIdentifier, "");
    }
}
else {
    /* throw new Estrel2EstudioException(
     * this.getName() + " no Type");
 */
170     } else {
        /* @param theXPath
         * @param theParser
         * @param theResult;
         */
        return result;
    }
}
180 /**
 * Parses a variable. <br>
 * Sideeffects: sets the class variables.
 */
190     /* Parses the XPath which leads to the Expression
      * @param theParser
      * @throws EstrelParserException
      *          if there is a parsing problem
      * @see kiel.fileInterface.estrel.EstrelParserException
      */
public final void parseEstrelISStatement(
    final EstrelParser theparser,
    final String theXPath)
    throws EstrelParserException {
    List anElementList = DOMHelper.getElements(theXPath);
    String theName = ((Element) anElementList.get(0));
    theparser.getDocument();
    theName = ((Element) anElementList.get(0))
        .getParentElement()
        .getFname();
}
200
210
220
230
240
250
    this.setFname(((Element) anElementList.get(0)).getParentElement()
        .getFname());
    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
                    .gettheEstrelModuleID((Element) anElementList.get(i))
                    .getAttributeValue("id");
        } else if (i == 2
            && ((Element) anElementList.get(i)).getName() != "NULL") {
            this.myExpression = new EstrelExpression(
                theXPath
                    + "[local-name(.) = "
                    + ((Element) anElementList.get(i)).getName()
                    + "] [oid='"
                    + ((Element) anElementList.get(i)).getFname()
                    + "']");
        } else if (i == 0) {
            /* @param theParser
             * @param theName
             */
            throw new EstrelParserException(
                "No Constantname");
        }
    }
}
*/
    /**
     * Returns the String representation of the variable.
     * @param anEstrelModule an estrel module.
     */
    public final String toString(final EstrelModule anEstrelModule) {
        String result = this.myConstantIdentifier;
        if (this.myExpression != null) {
            result += " := " + this.myExpression.toString(anEstrelModule);
        }
        if (this.myChannelType != null) {
            result += " : " + this.myChannelType;
        }
    }
}

```

C. Java Code für die Transformation von Esterel in SyncCharts

```
    }
    return result + "\n";
}
260
/***
 * @return Returns the myChannelType.
 */
public final String getChannelType() {
    return this.myChannelType;
}
270
*/
public final String getConstantIdentifier() {
    return this.myConstantIdentifier;
}
280
*/
public final String getExpression() {
    return this.myExpression;
}
*/
public final EsterelExpression getExpression() {
    return this.myExpression;
}
*/
public final String getTypeName() {
    return this.myChannelType.getTypeName();
}
*/
170
```

C.2.10. EstrelDeclaration

Der Klassenaufbau ist in der Abbildung C.2.10 dargestellt.

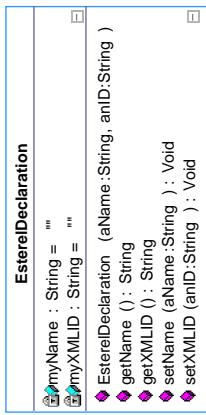


Abbildung C.8.: Klassendiagramm EstrelDeclaration

Auflistung C.15: Die Klasse EstrelDeclaration

```

package kiel.fileInterface.estrel.estrel2estudio;
/*
 * <p>
 * The superclass for all Estrel classes.
 * </p>
 */
10  * Copyright : Copyright (c) 2005
 * <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>
 * Company : Uni Kiel
 * </p>
 */
20  * the name.
 * the XML id in a file.
 */
private String myName = "";
public class EstrelDeclaration {
    /*
     * @return Returns the myName.
     */
    public final String getName() {
        return this.myName;
    }

    /*
     * @return Returns the myID.
     */
    private String myXMLID = "";
}

```

C. Java Code für die Transformation von Esterel in SyncCharts

```
60    public final String getXMLID() {
61        return this.myXMLID;
62    }
63
64    /**
65     * @param aName      The alname to set.
66     */
67    public final void setXMLID(
68        final String anID) {
69        this.myXMLID = anID;
70    }
71
72    /**
73     * @param anID      The anID to set.
74     */
75    public final void setAlname(
76        final String alname) {
77        this.myAlname = alname;
78    }
79
80}
```

C.2.11. EsterDeclarationSortByID

Auflistung C.16: Die Klasse EsterelDeclarationSortByID

```

    * @see kiel.fileInterface.estere12studio.EsterelDeclaration
  package kiel.fileInterface.estere12studio;
  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 standard constructor.
     */
    public EsterelDeclarationSortByID() {
      super();
    }
    /**
     * @param arg0
     * @param arg1
     * @return <p>
     * <math>\varphi_0</math> if <math>\arg_0 < \arg_1</math>, negative value if <math>\arg_0 > \arg_1</math>, positive
     * value if <math>\arg_1 < \arg_0</math>, negative value if <math>\arg_1 > \arg_0</math>, positive
     */
    public int compare(EsterelDeclaration arg0,
                      EsterelDeclaration arg1) {
      final Object arg0Obj = arg0;
      final Object arg1Obj = arg1;
      if (arg0Obj instanceof EsterelDeclaration
          && arg1Obj instanceof EsterelDeclaration) {
        return this.compare((EsterelDeclaration) arg0,
                           (EsterelDeclaration) arg1);
      }
      return 0;
    }
  }

```

C.2.12. EsterelDeclarationSortByName

Auflistung C.17: Die Klasse EsterelDeclarationSortByName

```

package kiel.fileInterface.estrel.estrel2estudio;
import java.util.Comparator;

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

    /**
     * @param arg0
     * @param arg1
     */
30    public int compare(
        Object arg0,
        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.

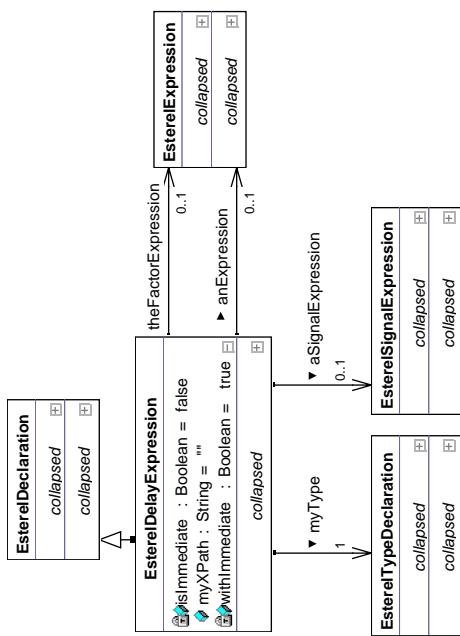


Abbildung C.9.: Klassendiagramm EsterelDelayExpression

Auflistung C.18: Die Klasse EsterDelayExpression

```
package kiel.fileInterface.estrel.estrel2studio;

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.estrel.EstrelParser;
import kiel.fileInterface.estrel.EstrelParserException;
import kiel.util.CompoundLabelException;
import kiel.util.DOMHelps;
import kiel.util.CompoundLabelParser;
import kiel.util.DOMLabelParser;

import org.json.Document;
import org.json.Element;
import org.json.JSONObject;
```

C. Java Code für die Transformation von Esterel in SyncCharts

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

C.2.14. EstrelExpression

Der Klassenaufbau ist in der Abbildung C.2.14 dargestellt.

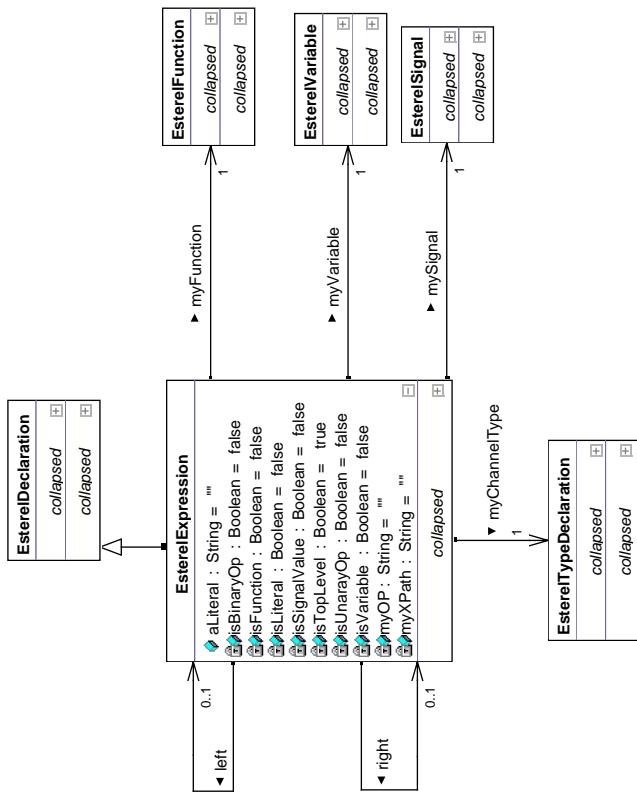


Abbildung C.10.: Klassendiagramm EstrelExpression

Auflistung C.19: Die Klasse EstrelExpression

```

import kiel.dataStructure.TransitionLabel;
import kiel.dataStructure.boolExp.BooleanExpression;
import kiel.dataStructure.intExp.IntegerExpression;
10 import kiel.fileInterface.estrel.EsterelParser;
import kiel.fileInterface.estrel.EsterelParserException;
import kiel.util.CompoundLabelException;
import kiel.util.CompoundLabelParser;
import kiel.util.DOMLabelParser;

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

import org.json.Document;
import org.json.Element;

/**
 * This class implements all esterel expressions .
 */
* <p>
* </p>
* <p>Copyright : Copyright (c) 2004
* </p>
* <p>Company : Uni Kiel
* </p>
* </p>
* <p>Author <a href="mailto:lk@informatik.uni-kiel.de">Lars Kuehl </a>
* </p>
* <p>Version $Revision: 1.33 $ last modified $Date: 2006/02/13 14:00:04 $
```

80

```

public class EsterelExpression extends EsterelDeclaration {
    /**
     * If the expression is a literals it stored as string.
     */
    private String aliteral;
```

90

```

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

100

```

    /**
     * If the expression is a function.
     */
    private booleanisFunction = false;
```

110

```

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

120

```

    /**
     * true, if the expression is a signal value.
     */
    private boolean isSignalValue;
```

130

```

    /**
     * <code> true </code>, if this expression is the root expression.
     */
    private boolean isTopLevel = true;
```

140

```

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

150

```

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

160

```

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

170

```

    /**
     * The id of the function.
     */
    private String myOP;
```

180

```

    /**
     * the operation as string.
     */
    private String mySignalId;
```

190

```

    /**
     * the signalid as string.
     */
    private String mySignal;
```

200

```

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

210

```

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

220

```

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

230

```

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

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

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

/**
 * Uses the standard constructor and parses an expression. <br>
 * Sideeffects: sets the class variables
 * @param theparser a estrel parser.
 * @param xpath the xpath to the expression in an xml document
 * @param xmlDoc the <code>org.jdom.Document</code> representation of an
 * exp file
 * @throws EstrelParserException
 * @see kiel.fileInterface.estrel.EstrelParserException
 */
public EstrelExpression(
    final EstrelParser theparser,
    final String xpath,
    final Document xmlDoc,
    final boolean topLevel)
throws EstrelParserException {
    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 (xpath != null) {
        // xmlDoc != null
        this.parseExpression(theparser,
            xpath,
            xmlDoc);
    }
}

/*
 * Converts the expression to a
 * <code>kiel.dataStructure.intexp.IntererExpression</code>.
 * @param anEstrelModule
 * @param anEstrelModule
 * @return a <code>kiel.dataStructure.intexp.IntererExpression</code>
 * @see kiel.dataStructure.TransitionLabel
 */
public final IntegerExpression convertToKiel(
    final EstrelModule anEstrelModule) {
    CompoundLabelParser getINSTANCE();
    // Setting Statechart for identifier searching
    CompoundLabelParser setStateChart(
        anEstrelModule.getEstrelStateChart());
    CompoundLabelParser setAllLocals(null,
        null);
    String aStringLabel = "("
        + this.tostring(anEstrelModule).trim() + ")";
    IntegerExpression anIntExp = null;
    try {
        anIntExp = CompoundLabelParser.parseIntExpression(aStringLabel);
    } catch (CompoundLabelException ex) {
        anIntExp = null;
    }
    return anIntExp;
}

/**
 * Uses the standard constructor and parses an expression. <br>
 * Sideeffects: sets the class variables
 * @param theparser a estrel parser.
 * @param xpath 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 EstrelParserException
 */
130
140
150
160
170
180
190
200
210
220
230
141
151
161
171
181
191
201
211
221
231

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

    }
}

240   /**
 * Converts the expression to a
 * <code>kiel.dataStructure.interp.IntegerExpression</code>.
 */
241   * @param anEsterelModule
 *   the esterel module
 * @param theLocalEvents
 *   all local events
 * @param theLocalVariables
 *   all local variables
 * @return a <code>kiel.dataStructure.interp.IntegerExpression</code>
 * @see kiel.dataStructure.TransitionLabel
 */
242   public final IntegerExpression convertToIntKiel(
243       final EsterelModule anEsterelModule,
244       final ArrayList<theLocalEvents>,
245       final ArrayList<theLocalVariables> {
246       CompoundLabelParser gerInstance();
247       // Setting StateChart for identifier searching
248       CompoundLabelParser setStatechartStateChart();
249       anEsterelModule.getEsterelStateChart();
250       CompoundLabelParser.setAllLocals(theLocalVariables,
251           theLocalEvents);
252       String aStringLabel = "!" +
253           + this.toString(anEsterelModule).trim() + "}";
254       IntegerExpression anIntExp = new IntegerExpression(aStringLabel);
255       try {
256           anIntExp = CompoundLabelParser.parseIntExpression(aStringLabel);
257       } catch (CompoundLabelException ex) {
258           anIntExp = null;
259       }
260       return anIntExp;
261   }

262   /**
 * Converts the expression to a
 * <code>kiel.dataStructure.floatExpression</code>.
 */
263   * @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
 */
264   public final FloatExpression convertToKielFloatExp(
265       final EsterelModule anEsterelModule,
266       final ArrayList<theLocalEvents>,
267       final ArrayList<theLocalVariables> {
268       CompoundLabelParser gerInstance();
269       // Setting StateChart for identifier searching
270       CompoundLabelParser.setStatechartStateChart();
271       anEsterelModule.getEsterelStateChart();
272       CompoundLabelParser.setAllLocals(theLocalVariables,
273           theLocalEvents);
274       String aStringLabel = "!" +
275           + this.toString(anEsterelModule).trim() + "}";
276       FloatExpression aFloatExp = new FloatExpression(aStringLabel);
277       try {
278           aFloatExp = CompoundLabelParser.parseFloatExpression(aStringLabel);
279       } catch (CompoundLabelException ex) {
280           aFloatExp = null;
281       }
282       return aFloatExp;
283   }

284   /**
 * Converts the expression to a
 * <code>kiel.dataStructure.doubleExp</code>.
 */
285   * @param anEsterelModule
 *   the esterel module
 * @param theLocalEvents
 *   all local events
 * @param theLocalVariables
 *   all local variables
 * @return aDoubleExp;
 */
286   public final DoubleExpression convertToKielDoubleExp(
287       final EsterelModule anEsterelModule,
288       final ArrayList<theLocalEvents>,
289       final ArrayList<theLocalVariables> {
290       CompoundLabelParser gerInstance();
291       // Setting StateChart for identifier searching
292       CompoundLabelParser.setStatechartStateChart();
293       anEsterelModule.getEsterelStateChart();
294       CompoundLabelParser.setAllLocals(theLocalVariables,
295           theLocalEvents);
296       String aStringLabel = "!" +
297           + this.toString(anEsterelModule).trim() + "}";
298       DoubleExpression aDoubleExp = null;
299       try {
300           aDoubleExp = CompoundLabelParser.parseDoubleExpression(aStringLabel);
301       } catch (CompoundLabelException ex) {
302           aDoubleExp = null;
303       }
304       return aDoubleExp;
305   }

306   /**
 * Converts the expression to a
 * <code>kiel.dataStructure.floatExpr.DoubleExpression</code>.
 */
307   * @param anEsterelModule
 *   the esterel module
 * @param theLocalEvents
 *   all local events
 * @param theLocalVariables
 *   all local variables
 * @return aFloatExpr;
 */
308   public final DoubleExpression convertToKielDoubleExpr(
309       final EsterelModule anEsterelModule,
310       final ArrayList<theLocalEvents>,
311       final ArrayList<theLocalVariables> {
312       CompoundLabelParser gerInstance();
313       // Setting StateChart for identifier searching
314       CompoundLabelParser.setStatechartStateChart();
315       anEsterelModule.getEsterelStateChart();
316       CompoundLabelParser.setAllLocals(theLocalVariables,
317           theLocalEvents);
318       String aStringLabel = "!" +
319           + this.toString(anEsterelModule).trim() + "}";
320       DoubleExpression aDoubleExpr = null;
321       try {
322           aDoubleExpr = CompoundLabelParser.parseDoubleExpression(aStringLabel);
323       } catch (CompoundLabelException ex) {
324           aDoubleExpr = null;
325       }
326       return aDoubleExpr;
327   }

328   /**
 * Converts the expression to a
 * <code>kiel.dataStructure.floatExpr.FloatExpression</code>.
 */
329   * @param anEsterelModule
 *   the esterel module
 * @param theLocalEvents
 *   all local events
 * @param theLocalVariables
 *   all local variables
 * @return aFloatExpr;
 */
330   public final DoubleExpression convertToKielFloatExpr(
331       final EsterelModule anEsterelModule,
332       final ArrayList<theLocalEvents>,
333       final ArrayList<theLocalVariables> {
334       CompoundLabelParser gerInstance();
335       // Setting StateChart for identifier searching
336       CompoundLabelParser.setStatechartStateChart();
337       anEsterelModule.getEsterelStateChart();
338   }
}

```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

    * 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 parser.getInstance();
        CompoundLabelParser.setIdentifierSearching(
            // Setting Statechart for identifier searching
            // Setting Statechart for identifier searching
            CompoundLabelParser.setStatechart(
                anEsterelModule.getIdentifierStatechart());
        CompoundLabelParser.setAllLocals(theLocalVariables,
            theLocalEvents);
        String aStringLabel = "{";
        + this.toString(anEsterelModule).trim() + "}";
        try {
            BooleanExpression aBoolExp = null;
            aBoolExp = CompoundLabelParser.parseBooleanExpression(aStringLabel);
        } catch (CompoundLabelException ex) {
            aBoolExp = null;
        }
        return aBoolExp;
    }

    /**
     * Converts the expression to a
     * <code>String</code>.
     * @param anEsterelModule
     * @return the esterel module
     * @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 xPath the Xpath which leads to the Expression
     * @param xmlDoc the <code>org.jdom.Document</code> representation of the
     *               XML file
     * @throws EsterelParserException
     * @see kiel.fileInterface.esterel.EsterelParserException
     */
    public final void parseExpression(

```

C. Java Code für die Transformation von Esterel in SyncCharts

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

C.2.15. EsterelFunctionDeclaration

Der Klassenaufbau ist in der Abbildung C.2.15 dargestellt.

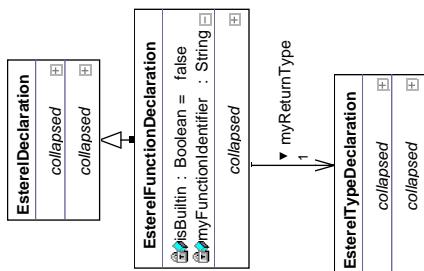


Abbildung C.11.: Klassendiagramm EsterelFunctionDeclaration

Aufstellung C.20: Die Klasse EsterelFunctionDeclaration

```

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;
    /**
     * <br>
     */
    private String myReturnString;

    public void setFunctionName(String name) {
        myFunctionIdentifier = name;
    }

    public void setReturnType(EsterelTypeDeclaration type) {
        myReturnType = type;
    }

    public void setReturnString(String str) {
        myReturnString = str;
    }

    public void setBuiltIn(boolean b) {
        isBuiltIn = b;
    }

    public void setVariable(EsterelVariable var) {
        inkIel = var;
    }

    public EsterelVariable getVariable() {
        return inkIel;
    }

    public boolean isBuiltIn() {
        return isBuiltIn;
    }

    public String getFunctionName() {
        return myFunctionIdentifier;
    }

    public EsterelTypeDeclaration getReturnType() {
        return myReturnType;
    }

    public String getReturnString() {
        return myReturnString;
    }
}

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

    * the value parameters.
40   */
41   private ArrayList myValueParameterTypes;
42   /**
43   * the simple constructor.
44   */
45   /**
46   */
47   public EsterelFunctionDeclaration() {
48     super();
49     /**
50      * <br>Sideeffects: class variables are changed.
51      * @param theparser the esterel parser
52      * @param theXPath the XPath to the EsterelVariable
53      * @throws EsterelParserException if something
54      * goes wrong at the parsing.
55      */
56     public EsterelFunctionDeclaration(
57       final EsterelParser theparser,
58       final String theXPath)
59     throws EsterelParserException {
60       super();
61       /**
62        * <br>Sideeffects: null;
63        * @param theparser the esterel parser
64        * @param theXPath to the EsterelVariable
65        * @throws EsterelParserException if something
66        * goes wrong at the parsing.
67        */
68       final EsterelParser theparser,
69       final String theXPath)
70     throws EsterelParserException {
71       /**
72        * <br>Sideeffects: null;
73        * @param theparser the esterel parser
74        * @param theXPath the XPath to the EsterelVariable
75        * @throws EsterelParserException if something
76        * goes wrong at the parsing.
77        */
78       /**
79        * Creates an EsterelFunction and fills the class variables.
80        * <br>Sideeffects: class variables are changed.
81        * @param theparser the esterel parser
82        * @param theXPath the XPath to the EsterelVariable
83        * @throws EsterelParserException if something
84        * goes wrong at the parsing.
85        */
86       /**
87        * Creates an EsterelFunction and fills the class variables.
88        * <br>Sideeffects: class variables are changed.
89        * @param theparser the esterel parser
90        * @param theXPath the XPath to the EsterelVariable
91        * @throws EsterelParserException if something
92        * goes wrong at the parsing.
93        */
94       /**
95        * Returns a variable representation of the esterel variable.
96        * @param anEsterelModule the esterel module which
97        * becomes a KIEL Statechart
98        * @param theLocalEvents Stack with the local events
99        * @param theLocalVariables Stack with all local variables
100       .convertToKiel(anEsterelModule,
101                     theLocalEvents,
102                     theLocalVariables);
103     }
104   }
105   /**
106   */
107   public final Variable convertToKiel(
108     final EsterelModule anEsterelModule,
109     final ArrayList theLocalEvents,
110     final ArrayList theLocalVariables)
111   throws Esterel2studioException {
112     if (this.intKiel == null) {
113       this.intKiel = new EsterelVariable();
114     }
115     this.intKiel.setVariableIdentifier(this.myFunctionIdentifier);
116     this.intKiel.setInOrOutEsterelProgram(true);
117     this.intKiel.setChannelType(this.myReturnType);
118   }
119   /**
120   */
121   public final String getFunctionIdentifier() {
122     return this.intKiel.getFunctionIdentifier();
123   }
124   /**
125   */
126   public final boolean isBuiltIn() {
127     return this.isBuiltIn;
128   }
129   /**
130   */
131   public final void parseEsterelStatement(
132     final EsterelParser theparser,
133     final String theXPath)
134   throws EsterelParserException {
135     List anElementList =
136       DOMHelpers.getElements(theXPath, theparser.getDocument());
137     int theEDVCounter = 0;
138     int i = 0;
139     this.setArrayList((Element) anElementList.get(0)).getRootElement()
140       .getFirstChild();
141     this.setXMLID((Element) anElementList.get(0)).getRootElement()
142       .getAttributeValue("id");
143     this.myReturnType = null;
144     if (this.getName().compareTo("BuiltInFunctionSymbol") == 0) {
145       this.isBuiltIn = true;
146       this.myFunctionIdentifier =
147         ((Element) anElementList.get(0)).getText();
148     } else {
149       this.myValueParameterTypes = new ArrayList();
150       while (i < anElementList.size()) {
151         Element anElement = (Element) anElementList.get(i);
152         if (anElement.getName().compareTo("EDV") == 0) {
153           theEDVCounter++;
154         } else if (i == 0) { // parse name
155           this.myFunctionIdentifier = anElement.getText();
156         }
157       }
158     }
159   }

```

```

    } else if (theEOVCounter == 0) {
        this.myValueParameterTypes.add(
            theparser
                .getEstrelModule(theparser.getactualEstrelModule()
                    .getEstrelTypeByID((Element) anElementList.get(i)))
        );
    } else if (theEOVCounter == 1) {
        this.myReturnType =
            theparser
                .getEstrelModule(theparser.getactualEstrelModule()
                    .getEstrelTypeByID((Element) anElementList.get(i)))
                .getAttributeValue("id");
    }
}
i++;
}
} //while
}
*/
/** 
 * Returns the String representation of the variable.
 */

```

170

```

    * @param anEstrelModule an <code>EstrelModule</code>
    * @return the String representation of an EstrelVariable
    */

```

```

public final String toString(
    final EstrelModule anEstrelModule) {

```

```

    if (this.isbuiltin) {
        return "";
    }

```

```

    String result = this.myFunctionIdentifier + " ";
    if (this.myValueParameterTypes != null) {

```

```

        for (int i = 0; i < this.myValueParameterTypes.size(); i++) {
            result += (String) this.myValueParameterTypes.get(i)
                + " ";
        }
    }

```

```

    result += " ) : " + this.myReturnType.getTypeName();
    return result;
}
}

```

180

```

    * Returns the String representation of the variable.
}
}

```

160

C.2.16. EsterelFunction

Aufistung C.21: Die Klasse EsterelFunction

```

package kiel.fileInterface.esterel.esterel2studio;

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;

10   /**
   * <p>
   * Represents a function call in esterel.
   * </p>
   */
20  * Copyright: Copyright (c) 2005
  * <p>
  * Company : Uni Kiel
  * </p>
  */
20  * @author <a href="mailto:ku@informatik.uni-kiel.de">Lars Kuehl </a>
  * @version $Revision: 1.6 $ last modified $Date: 2006/02/07 18:39:37 $
  */
20  public final class EsterelFunction
    extends EsterelDeclaration {
    /**
     * the functionexpressions .
     */
30    private EsterelExpression[] myExpressions;
    /**
     * the xml id of the functiondeclaration .
     */
30    private String myFunctionXMLID;
    /**
     * the function identifier .
     */
30    private String myFunctionId;
    /**
     * the return type .
     */
30    private EsterelTypeDeclaration myReturnType;
    /**
     * a constructor .
     */
30    * @param theparser a EsterelParser .
     * @param xpath the path to the function call
     * @param xmldoc the document
     * @throws EsterelParseException an exception
     */
30    public EsterelFunction( EsterelParser theparser,
                           final String xpath,
                           final Document xmldoc );
}

30  throws EsterelParserException {
super();
this.parseEsterelStatement(
    theparser,
    xpath,
    xmldoc);
}

60  /**
   * <p>
   * Returns the myFunctionId.
   * </p>
   */
60  public String getFunctionId() {
    /**
     * parses a function into the datastructure .
     */
60    return this.myFunctionId;
}

70  /**
   * Parses a function call in esterel.
   * <p>
   * @param theparser a EsterelParser .
   * @param xpath the xpath to the function.
   * @param xmldoc the document
   * @throws EsterelParserException
   * @throws EsterelParseException an exception
   */
70  public void parseEsterelStatement(
    final EsterelParser theparser,
    final String xpath,
    final Document xmldoc)
    throws EsterelParserException {
List anElementList = DOMHelpers.getElements(xpath,
                                             xmldoc);
final int ATTREE = 3;
if (anElementList.size() > 1) {
    if (((Element) anElementList.get(0)).getLocalName()
        .compareTo("NULL") != 0) {
        this.myReturntype =
            theparser
                .getEsterelModule(theparser.getActualEsterelModule())
                .getEsterelTypeID(((Element) anElementList.get(0)))
                .getActualEsterelModule(theparser
                    .getActualFunctionModule(theparser
                        .getFunctionID(this.myFunctionXMLID)));
    }
}
}

80  /**
   * <p>
   * Parses a function call in esterel.
   * </p>
   */
80  public void parseEsterelStatement(
    final EsterelParser theparser,
    final String xpath,
    final Document xmldoc)
    throws EsterelParserException {
List anElementList = DOMHelpers.getElements(xpath,
                                             xmldoc);
final int ATTREE = 3;
if (anElementList.size() > 1) {
    if (((Element) anElementList.get(0)).getLocalName()
        .compareTo("NULL") != 0) {
        this.myReturntype =
            theparser
                .getEsterelModule(theparser.getActualEsterelModule())
                .getEsterelTypeID(((Element) anElementList.get(0)))
                .getActualEsterelModule(theparser
                    .getActualFunctionModule(theparser
                        .getFunctionID(this.myFunctionXMLID)));
    }
}
}

90  /**
   * <p>
   * Parses a function call in esterel.
   * </p>
   */
90  public void parseEsterelStatement(
    final EsterelParser theparser,
    final String xpath,
    final Document xmldoc)
    throws EsterelParserException {
List anElementList = DOMHelpers.getElements(xpath,
                                             xmldoc);
final int ATTREE = 3;
if (anElementList.size() > 1) {
    if (((Element) anElementList.get(0)).getLocalName()
        .compareTo("NULL") != 0) {
        this.myReturntype =
            theparser
                .getEsterelModule(theparser.getActualEsterelModule())
                .getEsterelTypeID(((Element) anElementList.get(0)))
                .getActualEsterelModule(theparser
                    .getActualFunctionModule(theparser
                        .getFunctionID(this.myFunctionXMLID)));
    }
}
}

100 /**
   * <p>
   * Parses a function call in esterel.
   * </p>
   */
100 public void parseEsterelStatement(
    final EsterelParser theparser,
    final String xpath,
    final Document xmldoc)
    throws EsterelParserException {
List anElementList = DOMHelpers.getElements(xpath,
                                             xmldoc);
final int ATTREE = 3;
if (anElementList.size() > 1) {
    if (((Element) anElementList.get(0)).getLocalName()
        .compareTo("NULL") != 0) {
        this.myReturntype =
            theparser
                .getEsterelModule(theparser.getActualEsterelModule())
                .getEsterelTypeID(((Element) anElementList.get(0)))
                .getActualEsterelModule(theparser
                    .getActualFunctionModule(theparser
                        .getFunctionID(this.myFunctionXMLID)));
    }
}
}

```

```

        .getFunctionIdentifier();
    }
    if (anElementList.size() > aTHREE) {
        this.myExpressions = new EstrelExpression(anElementList.size() - aTHREE);
        for (int i = 2; i < anElementList.size() - 1; i++) {
            this.myExpressions[i - 2] =
                new EstrelExpression(
                    theparser,
                    "/*[@id=" +
                        ((Element) anElementList.get(i))
                            .getAttributeValue("id") +
                        ", */",
                        xmldoc);
        }
    } else {
        throw new EstrelParserException(
            "EstrelFunction: missing elements.");
    }
}

/**
 * @param anEstrelModule
 *      the module.
 * @return a string.
 */
130 public String toString(
    final EstrelModule anEstrelModule) {
String result = "";
EstrelFunctionDeclaration theDecl = anEstrelModule
    .getEstrelFunctionByID(this.myFunctionXMLID);
if (theDecl.isBuiltin()) {
    if (this.myExpressions.length == 1) {
        140 result = theFDcl.getFunctionIdentifier()
            + " "
            + this.myExpressions[0].toString(anEstrelModule);
    } else {
        result = this.myExpressions[0].toString(anEstrelModule)
            + " "
            + theFDcl.getFunctionIdentifier()
            + " "
            + this.myExpressions[1].toString(anEstrelModule);
    }
} else {
    result = theFDcl.getFunctionIdentifier();
    if (Estrel12EstudioProperties.getFunctions() {
        result += " ";
        if (this.myExpressions != null) {
            for (int i = 0; i < this.myExpressions.length - 1; i++) {
                150 result += this.myExpressions[i]
                    + Estrel12EstudioProperties
                        .getSeparatorString();
            }
            if (this.myExpressions.length > 1) {
                160 result += =
                    this.myExpressions[this.myExpressions.length - 1]
                        .toSString(anEstrelModule);
            }
        }
    }
}
170 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.estrel.estrel2estudio;           * the module functions.
50
/*
 * This class implements an estrel module.
 */
private EstrelFunctionDeclaration[] myFunctions = null;
10
/** the input and output signals */
private EstrelSignal[] myInputOutputSignals = null;
15
/** the input signals */
private EstrelSignal[] myInputSignals = null;
20
/** the output signals */
private EstrelSignal[] myOutputSignals = null;
25
/** the module procedures.
 */
private EstrelProcedureDeclaration[] myProcedures = null;
30
/*
 * the module relations.
 */
private EstrelRelation[] myRelations = null;
35
/*
 * the module returns.
 */
private EstrelSignal[] myReturnsSignals = null;
40
/*
 * the module sensors.
 */
private EstrelSignal[] mySensors = null;
45
/*
 * the module tasks.
 */
private EstrelTaskDeclaration[] myTasks = null;
50
/*
 * the module typs.
 */
private EstrelTypeDeclaration[] myTyps = null;
55
/*
 * Xpath which leads to a module.
 */
private String myXPath;
60
/** A representation of all estrel signals. */
65

```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

private EsterelSignal[] theEsterelSignals = null;
100    /** A representation of all esterel variables. */
private EsterelVariable[] theEsterelVariables = null;
101    //Public final void parseEsterelModule(final EsterelParser the
// parser){}
102    /**
103     * Simple constructor.
104     */
105    public EsterelModule() {
106        this.myXPath = "";
107    }
108    //public EsterelModule()
109    /**
110     * Parses a Document and set the class variables.
111     */
112    /**
113     * @param myModuleXPath
114     *      the xpath to the module
115     *      @throws EsterelParserException
116     *          if there are parsing errors
117     *      @see #parseEsterelModule(EsterelParser)
118     *      @see kiel.fileinterface.estrel.EsterelParser
119     */
120    public EsterelModule(
121        final String myModuleXPath)
122        throws EsterelParserException {
123        this();
124        this.myXPath = myModuleXPath
125        + "*/*[local-name()='Module']/*",
126    }
127    //public EsterelModule(final EsterelParser theparser) {
128    /**
129     * Simple constructor that sets the class variable
130     * <code>moduleName</code>
131     * and XPath to the module.
132     * @param name a name
133     * @param myModuleXPath the XPath to the module.
134     */
135    public EsterelModule(final String name,
136        final String myModulePath,
137        this.setname(name);
138        this.myXPath = myModulePath
139        + "*/*[local-name()='Module']/*";
140    }
141    /**
142     * adds a new signal to an <code>EsterelModule</code>
143     * @return the <code>EsterelSignal</code>
144     */
145    public final EsterelSignal addNewSignal() {
146        EsterelSignal result = new EsterelSignal();
147        if (this.theEsterelSignals.length > 0) {
148            result.setXMLID(this.theEsterelSignals[
149                this.theEsterelSignals.length - 1]
150                .getXMLID());
151        }
152        return result;
153    }
154    /**
155     * Adds a statement to the esterel module at index i.
156     * @param index
157     *      the index of the statement
158     *      @param aStatement
159     *          an <code>EsterelStatement</code>
160    */
161    public final void addStatement(int index, EsterelStatement aStatement) {
162        if (index < 0 || index > this.theEsterelSignals.length) {
163            throw new IndexOutOfBoundsException("Index " + index + " is out of bounds");
164        }
165        this.theEsterelSignals.add(index, aStatement);
166    }
167    /**
168     * Sorts the esterel module.
169     */
170    public void sort() {
171        Arrays.sort(this.theEsterelSignals,
172            new EsterelDeclarationSortByID());
173    }
174    /**
175     * adds a new signal to an <code>EsterelModule</code>
176     * @return the <code>EsterelSignal</code>
177     */
178    public final EsterelSignal addNewVariable() {
179        EsterelVariable result = new EsterelVariable();
180        result.setXMLID(this.theEsterelVariables[
181            this.theEsterelVariables.length - 1]
182                .getXMLID()
183                + "1");
184        return result;
185    }
186    /**
187     * @param name
188     */
189    public final void addNewVariable(String name) {
190        EsterelVariable result = new EsterelVariable();
191        result.setName(name);
192        result.setXMLID("1");
193        this.theEsterelVariables.add(result);
194    }
195    /**
196     * Sorts the esterel module.
197     */
198    public void sort() {
199        Arrays.sort(this.theEsterelVariables,
200            new EsterelDeclarationSortByID());
201    }
202    /**
203     * Sorts the esterel module.
204     */
205    public void sort() {
206        Arrays.sort(this.theEsterelVariables,
207            new EsterelDeclarationSortByIndex());
208    }

```

C. Java Code für die Transformation von Esterel in SyncCharts

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

    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);
    }
    alInitialArc.setSource(anInitialState);
    alInitialArc.setTarget(aState);
    root.setModule("Module");
    + this.getModuleName());
    root.addSubnode(alInitialArc);
    root.addSubnode(aState);
    this.anEstrelStateChart.setRootNode(root);
    aState.setParent(root);
    anInitialState.setParent(root);
    return this.anEstrelStateChart;
} /**
 * @param anID
 * @return the xmlId.
 */
330 public final ArrayList<EsterelConstant> getEstrelConstantByID(
    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;
}
340 /**
 * @param anID
 * @return the EstrelConstant with the id anID or null.
 */
341 public final EsterelConstant getEstrelConstantByID(
    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;
}
350 /**
 * @param anID a function name.
 * @return <code>EsterelFunctionDeclaration</code>
 */
351 public final EsterelFunctionDeclaration getEstrelFunctionByID(
    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;
}
360 /**
 * Returns the actual <code>anEstrelProgramm</code>.<br>
 * Sideffects: none
 */
361 public final ArrayList<EsterelTypeDeclaration> getEstrelTypeByName(
    final String aName) {
    EsterelTypeDeclaration result = null;
    for (int i = 0; i < this.myTypes.length;
        && result == null; i++) {
        if (this.myTypes[i].getXMLName() .compareTo(aName) == 0) {
            result = this.myTypes[i];
        }
    }
    return result;
}
370 /**
 * @param anEstrelProgramm</code>
 */
371 public final ArrayList<EsterelTypeDeclaration> getEstrelTypeByCode(
    final String code) {
    EsterelTypeDeclaration result = null;
    for (int i = 0; i < this.myTypes.length;
        && result == null; i++) {
        if (this.myTypes[i].getXMLName() .compareTo(code) == 0) {
            result = this.myTypes[i];
        }
    }
    return result;
}
380 /**
 * @param anID
 * @return the XMLID.
 */
381 public final String getEstrelTypeDeclarationID(
    final String anID) {
    EsterelTypeDeclaration dummy = new EsterelTypeDeclaration();
    dummy.setXMLID(anID);
    int key = Arrays.binarySearch(this.myTypes,
        dummy,
        new EsterelTypeDeclarationSortByID());
    if (key < 0) {
        return null;
    }
    return this.myTypes[key];
}
390 /**
 * @param anID
 * @return the XMLID.
 */
391 public final EsterelProcedureDeclaration getEstrelProcedureByID(
    final String anID) {
    EsterelProcedureDeclaration dummy = new EsterelProcedureDeclaration();
    dummy.setXMLID(anID);
    int key = Arrays.binarySearch(this.myProcedures,
        dummy,
        new EsterelProcedureDeclarationSortByID());
    if (key < 0) {
        return null;
    }
    return this.myProcedures[key];
}
400 /**
 * @param anID
 * @return the XMLID.
 */
401 public final EsterelTypeDeclaration getEstrelTypeByID(
    final String anID) {
    EsterelTypeDeclaration dummy = new EsterelTypeDeclaration();
    dummy.setXMLID(anID);
    int key = Arrays.binarySearch(this.myTypes,
        dummy,
        new EsterelTypeDeclarationSortByID());
    if (key < 0) {
        return null;
    }
    return this.myTypes[key];
}
410 /**
 * @param anID
 * @return the XMLID.
 */
411 public final EsterelDeclaration getEstrelDeclaration(
    final String anID) {
    EsterelDeclaration dummy = new EsterelDeclaration();
    dummy.setXMLID(anID);
    int key = Arrays.binarySearch(this.myDeclarations,
        dummy,
        new EsterelDeclarationSortByID());
    if (key < 0) {
        return null;
    }
    return this.myDeclarations[key];
}
420 /**
 * @param aName a type name.
 * @return <code>EsterelTypeDeclaration</code>
 */
421 public final EsterelTypeDeclaration getEstrelTypeByName(
    final String aName) {
    EsterelTypeDeclaration result = null;
    for (int i = 0; i < this.myTypes.length;
        && result == null; i++) {
        if (this.myTypes[i].getXMLName() .compareTo(aName) == 0) {
            result = this.myTypes[i];
        }
    }
    return result;
}
430 /**
 * Returns the actual <code>anEstrelProgramm</code>.<br>
 * Sideffects: none
 */
431 public final ArrayList<EsterelTypeDeclaration> getEstrelTypeByCode(
    final String code) {
    EsterelTypeDeclaration result = null;
    for (int i = 0; i < this.myTypes.length;
        && result == null; i++) {
        if (this.myTypes[i].getXMLName() .compareTo(code) == 0) {
            result = this.myTypes[i];
        }
    }
    return result;
}

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

490   * @param aName      * @return <code>EsterelSignal</code>
491   */
492   public final EsterelSignal getEsterelSignalByOriginalName(
493       final String aName) {
494       final EsterelSignal result = null;
495       for (int i = 0; i < this.theEsterelSignals.length
496           && result == null; i++) {
497           if (this.theEsterelSignals[i].getOriginalSignalIdentifier()
498               .compareTo(aName) == 0) {
499               result = this.theEsterelSignals[i];
500           }
501       }
502       return result;
503   }

504   /**
505    * @param anID      * @param aName      * @return <code>the xmId.
506    * @return the EsterelRelationDeclaration with the id anID or null.
507    */
508   public final EsterelRelation getEsterelRelationByID(
509       final String anID) {
510       final EsterelRelation dummy = new EsterelRelation();
511       dummy.setXMLID(anID);
512       int key = Arrays.binarySearch(this.myRelations,
513           dummy,
514           new EsterelDeclarationSortByID());
515       if (key < 0) {
516           return null;
517       }
518       return this.myRelations[key];
519   }

520   /**
521    * @param anID      * @param aName      * @return <code>the xmId.
522    * @return the EsterelTaskDeclaration with the id anID or null.
523    */
524   public final EsterelTaskDeclaration getEsterelTaskByID(
525       final String anID) {
526       final EsterelTaskDeclaration dummy = new EsterelTaskDeclaration();
527       dummy.setXMLID(anID);
528       int key = Arrays.binarySearch(this.myTasks,
529           dummy,
530           new EsterelDeclarationSortByID());
531       if (key < 0) {
532           return null;
533       }
534       return this.myTasks[key];
535   }

536   /**
537    * @param anID      * @param aName      * @return <code>the xmId.
538    * @return the EsterelSignal with the id anID or null.
539    */
540   public final EsterelSignal getEsterelSignalByID(
541       final String anID) {
542       final EsterelSignal dummy = new EsterelSignal();
543       dummy.setXMLID(anID);
544       int key = Arrays.binarySearch(this.theEsterelSignals,
545           dummy,
546           new EsterelDeclarationSortByID());
547       if (key < 0) {
548           return null;
549       }
550       return this.theEsterelSignals[key];
551   }

552   /**
553    * @param aName      * @return <code>EsterelVariable</code>
554    */
555   public final EsterelVariable getEsterelVariableByOriginalName(
556       final String aName) {
557       EsterelVariable result = null;
558       for (int i = 0; i < this.theEsterelVariables.length
559           && result == null; i++) {
560           if (this.theEsterelVariables[i].getOriginalVariableIdentifier()
561               .compareTo(aName) == 0) {
562               result = this.theEsterelVariables[i];
563           }
564       }
565       return result;
566   }

567   /**
568    * @param aName      * @return <code>the xmId.
569    * @return an esterel module and set values. <br>
570    * Sideeffects: see at subclasses
571    */
572   /**
573    * @param aParser
574    */

```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

* * an instance 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 NodeName
    List<Element> anElementList = DOMHelpers.getElements(this.myXPath,
        theparser.getXMLDocument());
    String xpath = "" + "[@Id='"
        + ((Element) anElementList.get(0)).getAttributeValue("id") + "'";
    anElementList = DOMHelpers.getElements(anXPath,
        theparser.getXMLDocument());
    this.setNodeName((Element) anElementList.get(0)).getText();
    //get type
    anXPath = this.myXPath
        + "[@Local-name(.)='SymbolTable']/*"
        + "[@Local-name(.)='TypeSymbol', "
        + "or Local-name(.)='BuiltInTypeSymbol']/*";
    anElementList = DOMHelpers.getElements(anXPath,
        theparser.getXMLDocument());
    Arraylist<String> theTyps = new ArrayList();
    for (int i = 0; anElementList.size() > i; i++) {
        Element anElement = ((Element) anElementList.get(i));
        EsterelTypeDeclaration atype =
            new EsterelTypeDeclaration(
                anElement.getAttribute("name"),
                anElement.getAttribute("type"),
                anElement.getText());
        theTyps.add(atype);
    }
    theTyps.trimToSize();
    this.myTyps =
        (EsterelTypeDeclaration[])
        theTyps.toArray(new EsterelTypeDeclaration[theTyps.size()]);
    Arrays.sort(this.myTyps,
        new EsterelDeclarationSortByID());
    // get functions
    anXPath = this.myXPath
        + "[@Local-name(.)='SymbolTable']/*"
        + "[@Local-name(.)='FunctionSymbol', "
        + "or Local-name(.)='BuiltInFunctionSymbol']/*";
    anElementList = DOMHelpers.getElements(anXPath,
        theparser.getXMLDocument());
    Arraylist<String> theFunctions = new ArrayList();
    for (int i = 0; i < anElementList.size(); i++) {
        Element anElement = ((Element) anElementList.get(i));
        theFunctions.add(new EsterelFunctionDeclaration(
            anElement.getAttribute("name"),
            anElement.getAttribute("type"),
            anElement.getAttribute("value")));
    }
    theFunctions.trimToSize();
    this.myFunctions =
        (EsterelFunctionDeclaration[])
        theFunctions.toArray(new EsterelFunctionDeclaration[theFunctions.size()]);
    Arrays.sort(this.myFunctions,
        new EsterelDeclarationSortByID());
    // get constants
    anXPath = this.myXPath
        + "[@Local-name(.)='SymbolTable']/*"
        + "[@Local-name(.)='ConstantSymbol']/*";
    anElementList = DOMHelpers.getElements(anXPath,
        theparser.getXMLDocument());
    Arraylist<String> theConstants = new ArrayList();
    for (int i = 0; anElementList.size() > i; i++) {
        Element anElement = ((Element) anElementList.get(i));
        theConstants.add(new EsterelConstant(
            anElement.getAttribute("name"),
            anElement.getAttribute("value")));
    }
    theConstants.trimToSize();
    this.myConstants =
        (EsterelConstant[])
        theConstants.toArray(new EsterelConstant[theConstants.size()]);
    // get procedures
    anXPath = this.myXPath
        + "[@Local-name(.)='ProcedureSymbol']/*";
    anElementList = DOMHelpers.getElements(anXPath,
        theparser.getXMLDocument());
    Arraylist<String> theProcedures = new ArrayList();
    for (int i = 0; i < anElementList.size(); i++) {
        Element anElement = ((Element) anElementList.get(i));
        theProcedures.add(new EsterelProcedureDeclaration(
            anElement.getAttribute("name"),
            anElement.getAttribute("type"),
            anElement.getAttribute("value")));
    }
    theProcedures.trimToSize();
    this.myProcedures =
        (EsterelProcedureDeclaration[])
        theProcedures.toArray(new EsterelProcedureDeclaration[theProcedures.size()]);
    Arrays.sort(this.myProcedures,
        new EsterelDeclarationSortByID());
    // get tasks
    anXPath = this.myXPath
        + "[@Local-name(.)='TaskSymbol']/*";
    anElementList = DOMHelpers.getElements(anXPath,
        theparser.getXMLDocument());
    Arraylist<String> theTasks = new ArrayList();
    for (int i = 0; i < anElementList.size(); i++) {
        Element anElement = ((Element) anElementList.get(i));
        theTasks.add(new EsterelTaskDeclaration(
            anElement.getAttribute("name"),
            anElement.getAttribute("type"),
            anElement.getAttribute("value")));
    }
    theTasks.trimToSize();
    this.myTasks =
        (EsterelTaskDeclaration[])
        theTasks.toArray(new EsterelTaskDeclaration[theTasks.size()]);
    Arrays.sort(this.myTasks,
        new EsterelDeclarationSortByID());
}

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

660      "/*@id=" +
        + ((Element) anElementList.get(1))
        .getAttributeruleValue("id") +
        + "]/*");
    }
    // for
    theTasks.trimToSize();
    this.myTasks =
        (EsterelTaskDeclaration[]) theTasks
        .toarray(new EsterelTaskDeclaration[theTasks.size()]);
    Arrays.sort(this.myTasks,
        new EsterelDeclarationSortByID());
    670    // 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.myPath +
        "[local-name(.) = 'SignalSymbol']",
        theparser.getXMLDocument());
    allVariables.addAll(DOMHelpers
        .getElements(this.myPath +
        "[local-name(.) = 'VariableSymbol']",
        theparser.getXMLDocument()));
    680    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));
    }
    690    Arrays.sort(this.theEsterelVariables,
        new EsterelDeclarationSortByID());
    for (int i = 0; i < this.theEsterelVariables.length; 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);
        }
    }
    700    // for
    theparser = this.myXPath
        + "[local-name(.) = " +
        + ((Element) anElementList.get(1)).getURIName() +
        + "[@id=" +
        + "]*]/*[@id=" +
        + "]*";
    anElementList = DOMHelpers.getElements(anXPath,
        theparser.getXMLODocument());
    for (int i = 0; i < anElementList.size(); i++) {
        theRelations.add(new EsterelRelation(
            theparser,
            this.myXPath
                + "[local-name(.) = " +
                + ((Element) anElementList.get(i)).getURIName() +
                + "[@id=" +
                + "]*]/*[@id=" +
                + "]*");
    }
    710    // for
    theparser = this.myXPath
        + "[local-name(.) = 'Implication']";
    anElementList = DOMHelpers.getElements(anXPath,
        theparser.getXMLODocument());
    for (int i = 0; i < anElementList.size(); i++) {
        theRelations.add(new EsterelRelation(
            theparser,
            this.myXPath
                + "[local-name(.) = " +
                + ((Element) anElementList.get(i)).getURIName() +
                + "[@id=" +
                + "]*]/*[@id=" +
                + "]*");
    }

```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

770           + " [not(local-name(.)='NULL')]");
    }
    //for
    theRelations.trimToSize();
    this.myRelations = (EstrelRelation[]) theRelations
        .toarray(new EstrelRelation[theRelations.size()]);
    Arrays.sort(this.myRelations,
        new EstrelDeclarationSortByID());
}
//public parse

/***
 * Returns a string representation of an estrel module.
 *
 * @return a String representation of an estrel statement
 * @see kiel.fileInterface.estrel.estrel2estudio.EstrelModule
 */
public final String toString() {
    String aModuleString = "Module "
        + this.getName() + ":" + "\n";
    String aInputString = "";
    String aInputOutputString = "";
    String aInputOutputString = "";
    if (this.myTypes.length > 0) {
        aModuleString += " type ";
    }
    for (int i = 0; i < this.myTypes.length; i++) {
        aModuleString += this.myTypes[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";
    }
}
//for
theRelations.trimToSize();
this.myRelations = (EstrelRelation[]) theRelations
    .toarray(new EstrelRelation[theRelations.size()]);
Arrays.sort(this.myRelations,
    new EstrelDeclarationSortByID());
}
//public parse

/***
 * Returns a string representation of an estrel module.
 *
 * @return a String representation of an estrel statement
 * @see kiel.fileInterface.estrel.estrel2estudio.EstrelModule
 */
public final String toString() {
    String aModuleString = "Module "
        + this.getName() + ":" + "\n";
    String aInputString = "";
    String aInputOutputString = "";
    String aInputOutputString = "";
    if (this.myTypes.length != 0) {
        for (int i = 0; i < this.myInputSignals.length; i++) {
            aInputString += this.myInputSignals[i].toString(this)
                + " \n";
        }
        if (this.myOutputSignals.length != 0) {
            for (int i = 0; i < this.myOutputSignals.length; i++) {
                aOutputString += this.myOutputSignals[i].toString(this)
                    + " \n";
            }
            if (this.myInputOutputSignals.length != 0) {
                for (int i = 0; i < this.myInputOutputSignals.length; i++) {
                    aInputOutputString += this.myInputOutputSignals[i].toString(this)
                        + " \n";
                }
                if (this.myInputOutputSignals.length != 0) {
                    for (int i = 0; i < this.myInputOutputSignals.length; i++) {
                        aInputOutputString += this.myInputOutputSignals[i].toString(this)
                            + " \n";
                    }
                    aModuleString += "OutputSignals ";
                    aModuleString += aInputOutputString
                        + " \n";
                }
                if (this.myInputOutputSignals.length != 0) {
                    for (int i = 0; i < this.myInputOutputSignals.length; i++) {
                        aInputOutputString += this.myInputOutputSignals[i].toString(this)
                            + " \n";
                    }
                    aModuleString += "InputSignals ";
                    aModuleString += aInputString
                        + " \n";
                }
            }
            aModuleString += "OutputSignals ";
            aModuleString += aInputOutputString
                + " \n";
        }
        if (this.mySensors.length > 0) {
            for (int i = 0; i < this.mySensors.length; i++) {
                aModuleString += this.mySensors[i].toString(this)
                    + " \n";
            }
            if (this.myRelations.length > 0) {
                for (int i = 0; i < this.myRelations.length; i++) {
                    aModuleString += this.myRelations[i].toString(this)
                        + " \n";
                }
            }
            aModuleString += "(EstrelStatement) this.anEstrelProgram
                .get(0))";
            aModuleString += " relations ";
        }
        for (int i = 0; i < this.myRelations.length; i++) {
            aModuleString += " end module "
                + this.getModuleName();
        }
        return aModuleString;
    }
}
/***
 * @return Returns the aEstrelStateChart .
 */
public final StateChart getEstrelStateChart() {
    return this.anEstrelStateChart;
}

```

C. Java Code für die Transformation von Esterel in SyncCharts

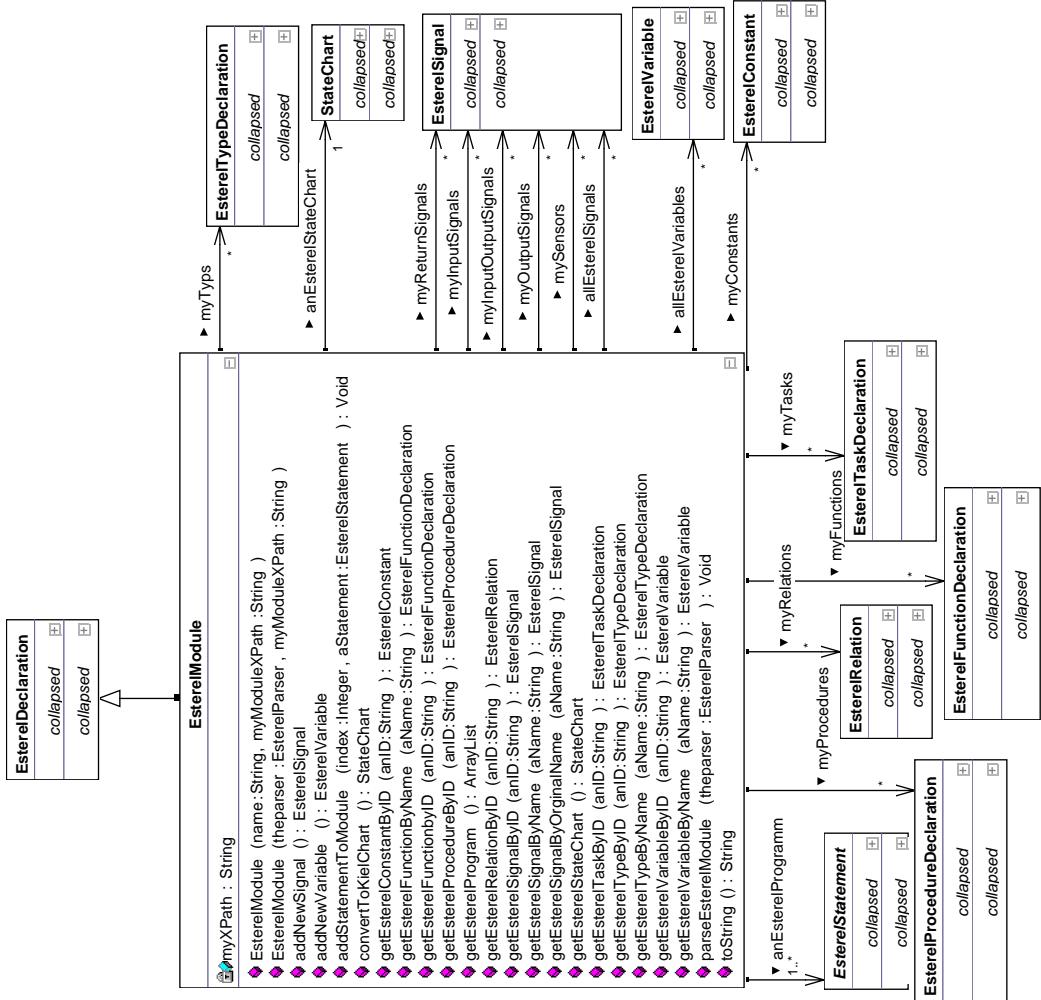


Abbildung C.12.: Klassendiagramm EsterelModule

C.2.18. EstrelProcedureDeclaration

Der Klassenaufbau ist in der Abbildung C.2.18 dargestellt.

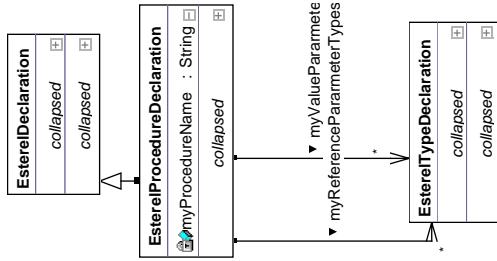


Abbildung C.13.: Klassediagramm EstrelProcedureDeclaration

Auflistung C.23: Die Klasse EstrelProcedureDeclaration

```

package kiel.fileInterface.estrel.estrel2estudio;
import java.util.ArrayList;
import java.util.List;
import kiel.util.DOHelpers;
import kiel.fileInterface.estrel.EstrelParser;
import kiel.fileInterface.estrel.EstrelParserException;
import org.json.Document;
import org.json.Element;
/*
 * <p> It represents estrel procedures .</p>
 */
10   /**
 * <p>Copyright: Copyright (c) 2005</p>
 * <p>Company: Uni Kiel</p>
 */
30   */
* the value parameters.
 */
  
```

This code listing shows the Java implementation of the **EstrelProcedureDeclaration** class. It includes imports for `kiel.fileInterface.estrel.estrel2estudio`, `java.util.ArrayList`, `java.util.List`, `kiel.util.DOHelpers`, `kiel.fileInterface.estrel.EstrelParser`, `kiel.fileInterface.estrel.EstrelParserException`, `org.json.Document`, and `org.json.Element`. The class contains several annotations and comments, including a multi-line string at the bottom. Line numbers 10 and 30 are present in the code.

C. Java Code für die Transformation von Esterel in SyncCharts

```

/*
 * private ArrayList myValueParameterTypes;
 */
 * the simple constructor.
 */
 * public EsterelProcedureDeclaration() {
super();
this.myReferenceParameterTypes = null;
this.myValueParameterTypes = null;
this.myProcedureName = "";
}

 /**
 * Creates an EsterelProcedure and fills the class variables.
 * <br>Sideeffects: class variables are changed.
 * @param theXPath the XPath the esterel parser
 * @param theEPath the EPath to the EsterelVariable
 * @throws EsterelParserException if something
 * goes wrong at the parsing.
 */
public EsterelProcedureDeclaration(
final EsterelParser theparser,
final String theXPath)
throws EsterelParserException {
super();
this.myReferenceParameterTypes = null;
this.myValueParameterTypes = null;
if (theXPath != null && theparser != null) {
this.parseEsterelStatement(theparser, theXPath);
} else {
throw new EsterelParserException(
"ProcedureDeclaration : no parser or path");
}
}

 /**
 * Returns the type of the DelayExpression.
 * <br> Sideeffects: none
 * @param delayList the children elements of the DelayExpression
 * @param anElementIndex the number of the delayElement in aDelayList
 * @param xmlDoc the <code>id</code>.Document</code>
 * representation of the .exp file
 * @return the type name
 * @throws EsterelParserException if no type is found.
 */
private String getType(
final List delayList,
final int anElementIndex,
final Document xmlDoc)
throws EsterelParserException {
if ((Element) delayList.get(anElementIndex)).getTagName() == "Ref" {
String aRefXPathSearchString =
"//*[@id=" + (
(Element) delayList.get(anElementIndex)).getAttribute("value")
+ "]";
}
}

```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

C.2.19. EsterelRelation

Der Klassenaufbau ist in der Abbildung C.2.19 dargestellt.

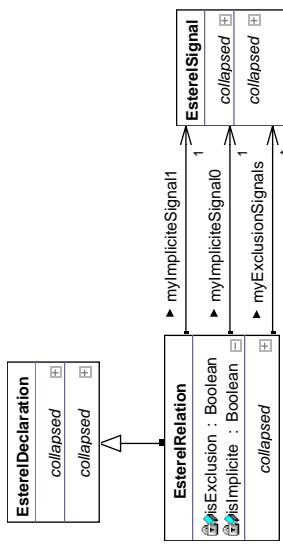


Abbildung C.14: Klassediagramm EsterelRelation

Auflistung C.24: Die Klasse EsterelRelation

```

package kiel.fileInterface.estrel.estrel2studio;
public class EstrelRelation extends EstrelDeclaration {
    /**
     * true if is explicite.
     */
    private boolean isExclusion;
    /**
     * true if is implicite.
     */
    private boolean isImplicit;
    /**
     * the explicite signals.
     */
    private ArrayList myExclusionSignals;
    /**
     * the implicite signal 0.
     */
    private String myImplicitSignal0;
    /**
     * the implicite signal 1.
     */
    private String myImplicitSignal1;
    /**
     * It represents estrel Relations .
     */
    private Company Uni Kiel
    /**
     * Copyright : Copyright (c) 2005
     */
    /**
     * $Revision: 1.19 $ last modified $Date: 2006/02/06 18:35:29 $
     * $Author: kuehl <kuh-mailto:luwinform@uni-kiel.de> Lars Kuehl </a>
     * @version $Revision: 1.19 $ last modified $Date: 2006/02/06 18:35:29 $
     * @version $Revision: 1.19 $ last modified $Date: 2006/02/06 18:35:29 $

```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

    /**
     * the simple constructor.
     */
    public EsterelRelation() {
        super();
        this.isExclusion = false;
        this.isImplicitite = false;
        this.myImplicititeSignal0 = "";
        this.myImplicititeSignal1 = "";
        this.myExclusionSignals = null;
    }

    /**
     * Creates an EsterelRelation and fills the class variables. <br>
     * Sideeffects: class variables are changed.
     */
    * @param theparser
    * @param theXPath
    * @throws EsterelParserException
    * @throws EsterelVariableException
    * @throws EsterelRelationException
    */
    public EsterelRelation(final EsterelParser theparser, final String theXPath)
        throws EsterelParserException {
        super();
        this.isExclusion = false;
        this.isImplicitite = false;
        this.myImplicititeSignal0 = null;
        this.myImplicititeSignal1 = null;
        this.myExclusionSignals = null;
        if (theXPath != null && theparser != null) {
            this.parseEsterelStatement(theparser, theXPath);
        } else {
            throw new EsterelParserException("RelationDeclaration"
                + " : no parser or path");
        }
    }
}

/**
 * Returns the signal of the Signalexpression. <br>
 */
* @param anElement
* @param anXMLdoc
* @throws EsterelParserException
*/
private String getSignal(final Element anElement, final Document xmldoc)
    throws EsterelParserException {
    if (anElement.getName() .compareTo("Ref") == 0) {
        return anElement.getText();
    } else {
        String abitPathSearchString = /* */ "[@id='"
            + anElement.getAttributeValue("id") +
            "+']/*[not(@local-name(.)='BOV')]" +
            "+[@local-name(.)='NULL'])";
        List theRefElementList = DOMHelpers.getElements(
            abitPathSearchString,
            xmldoc);
        if (theRefElementList.size() > 0
            && ((Element) theRefElementList.get(1)).getParentElement() != null) {
            return ((Element) theRefElementList.get(0)).getText();
        }
        throw new EsterelParserException("EsterelRelation: found no signal");
    }
}

/**
 * Parses a Relation. <br>
 * Sideeffects: sets the class variables.
 */
* @param theXPath
* @param theparser
* @throws EsterelParserException
* @throws EsterelVariableException
* @throws EsterelRelationException
* @see kiel.fileInterface.EsterelParserException
*/
public final void parseEsterelStatement(
    final EsterelParser theparser,
    final String theXPath) throws EsterelParserException {
    List anElementList = DOMHelpers.getElements(theXPath, theparser
        .getXMIDocument());
    Element anElement = (Element) anElementList.get(0);
    if (anElement.getTagName() .compareTo("Implication") == 0) {
        this.myImplicititeSignal0 = this.getSignal(anElement, theparser
            .getXMIDocument());
        this.myImplicititeSignal1 = this.getSignal(anElement, theparser
            .getXMIDocument());
        this.isImplicitite = true;
    } else if (anElement.getParentElement() .getTagName()
        .compareTo("Exclusion") == 0) {
        this.isExclusion = true;
        this.myExclusionSignals = new ArrayList();
        for (int i = 0; i < anElementList.size(); i++) {
            anElement = (Element) anElementList.get(i);
            this.myExclusionSignal1 = this.getSignal(anElement, theparser
                .getXMIDocument());
        }
    }
}

/**
 * Returns the signals name
 */
* @param anElement
* @param anXMLdoc
* @throws EsterelParserException
*/
private String getSignal(final Element anElement, final Document xmldoc)
    throws EsterelParserException {
    if (anElement.getName() .compareTo("Ref") == 0) {
        return anElement.getText();
    } else {
        String abitPathSearchString = /* */ "[@id='"
            + anElement.getAttributeValue("id") +
            "+']/*[not(@local-name(.)='BOV')]" +
            "+[@local-name(.)='NULL'])";
        List theRefElementList = DOMHelpers.getElements(
            abitPathSearchString,
            xmldoc);
        if (theRefElementList.size() > 0
            && ((Element) theRefElementList.get(1)).getParentElement() != null) {
            return ((Element) theRefElementList.get(0)).getText();
        }
        throw new EsterelParserException("EsterelRelation: found no signal");
    }
}

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

*param anEsterelModule an <code>EsterelModule</code>
 * @return the String representation of an EsterelVariable
 */
public final String toString(final EsterelModule anEsterelModule) {
    170    String result = "";
    if (this.isImplicit) {
        result = this.myImplicitSignal0 + " => " + this.myImplicitSignal1;
    } else if (this.isExclusion) {
        180
        for (int i = 0; i < this.myExclusionSignals.size() - 1; i++) {
            result += (String) this.myExclusionSignals.get(i) + " # ";
        }
        result += (String) this.myExclusionSignals
            .get(this.myExclusionSignals.size() - 1);
    }
    return result;
}

```

C.2.20. EstrelSignalExpression

Der Klassenaufbau ist in der Abbildung C.2.20 dargestellt.

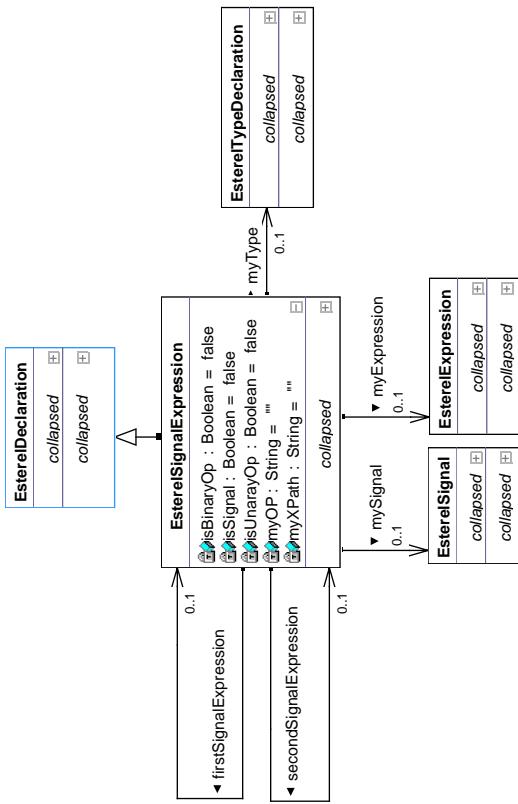


Abbildung C.15.: Klassendiagramm `EstrelSignalExpression`

Auflistung C.25: Die Klasse `EstrelSignalExpression`

```

import kiel.util.DOMHelpers;
import org.json.Document;
import org.json.Element;

/*
 * Represents a SignalExpression in estrel.
 */
20
 * <p>
 * </p>
 * Copyright : Copyright (c) 2004
 * <p>
 * </p>
 * Company : Uni Kiel
 */

package kiel.fileInterface.estrel.estrel2estudio;

import java.util.ArrayList;
import java.util.List;
import kiel.dataStructure.CompoundLabel;
import kiel.dataStructure.StringLabel;
import kiel.dataStructure.TransitionLabel;
import kiel.dataStructure.BooleanExpression;
import kiel.fileInterface.estrel.EsterelParser;
import kiel.fileInterface.estrel.EsterelParserException;
import kiel.util.CompoundLabelException;
import kiel.util.CompoundLabelParser;
  
```

10

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

    * @see kiel.dataStructure.Transitionlabel
    */
    public final Transitionlabel convertToKiel(
        final EstrelModule anEstrelModule,
        final ArrayList<Transitionlabel> theLocalEvents,
        final ArrayList<String> theLocalVariables,
        CompoundLabelParser getInstance() {
            // Setting StratChart for identifier searching
            CompoundLabelParser.setStateChart(anEstrelModule
                .getEstrelInstanceStateChart());
        }
    )
    CompoundLabelParser setAllLocals(
        theLocalVariables,
        .parseBooleanExpression(aStringLabel);
    )
    String aStringLabel = this.toString(anEstrelModule).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);
    }
    // convert alabel;
    return alabel;
}
/***
 * Returns the operation of the Signalexpression. <br>
 * Sideeffects: none
 * @param anElementList
 * @return the children elements of the SignalExpression
 * @throws EstrelParserException
 *          if no operation is found.
 */
private String getOperation(
    final List<anElementList>
)
throws EstrelParserException {
    if (((Element) anElementList.get(1)).getName() == "S") {
        return ((Element) anElementList.get(1)).getText();
    }
}
*/
/* Returns an <code>EstrelSignalExpression</code> of a sub
 */

```

```

250     * SignalExpression. <br>
251     * Sideeffects: none
252     *
253     * @param anElementList the children elements of the SignalExpression
254     * @param theparser an<code>EsterelParser</code>
255     * @param xmldoc the <code>org.jdom.Document</code> representation of the
256     * children of the SignalExpression
257     * @throws EsterelParserException
258     * @throws org.kiel.fileInterface.EsterelParserException
259     * @throws exp file
260     * is only delivered
261     * @see kiel.fileInterface.EsterelParserException
262     * @return a sub SignalExpression
263     */
264     private EsterelSignalExpression getSignalExpression(
265         final EsterelParser theparser,
266         final List anElementList,
267         final Document xmldoc)
268         throws EsterelParserException {
269     final int firstIndex = 2;
270     final int secondIndex = 3;
271     int actualElementNumber = 0;
272     if (this.firstSignalExpression == null) {
273         actualElementNumber = firstIndex;
274     } else if (this.secondSignalExpression == null) {
275         actualElementNumber = secondIndex;
276     } else {
277         actualElementNumber = -1;
278     }
279     if (actualElementNumber == firstIndex) {
280         return new EsterelSignalExpression(
281             theparser,
282             this.myXPath
283                 + "[local-name(.)=" +
284                 + "get(actualElementNumber).getName()"
285                 + "]@[id=" +
286                 + "((Element) anElementList"
287                 + ".get(actualElementNumber)"
288                 + ".getAttributeValue('id') + ")]/*",
289             xmldoc);
290     } else if (actualElementNumber == secondIndex) {
291         return new EsterelSignalExpression(
292             theparser,
293             this.myXPath
294                 + "[local-name(.)=" +
295                 + "get(actualElementNumber).getName()"
296                 + "]@[id=" +
297                 + "((Element) anElementList"
298                 + ".get(actualElementNumber)"
299                 + ".getAttributeValue('id') + ")]/*",
300             xmldoc);
301     } else {
302         return null;
303     }
304 }
305
306 /**
307  * Parses a SignalExpression. <br>
308  * Sideffects: sets the class variables
309  * @param theparser an esterel parser.
310  * @param searchXPath the XPath which leads to the Signalexpression
311  * @param xmldoc the <code>org.jdom.Document</code> representation of
312  * the exp file
313  * @throws EsterelParserException
314  * @throws org.kiel.fileInterface.EsterelParserException
315  */
316 public final void parseSignalExpression(
317     final EsterelParser theparser,
318     final String searchXPath,
319     final Document xmldoc)
320     throws EsterelParserException {
321     int i = 0;
322     this.myXPath = searchXPath;
323     List anElementList = DOMHelpers
324         .getElements(
325             this.myXPath
326                 + "[not(local-name(.)='ENV')] [not(local-name(.)='NULL')]"
327                 + "/*",
328                 xmldoc);
329     // the XPath finds the children of the SignalExpression.
330     String actualElement = ((Element) anElementList.get(i))
331         .getTagName();
332     if (actualElement == "UnaryOp") {
333         this.isUnaryOp = true;
334         this.myType =
335             theparser
336                 .getEsterelModule(theparser.getActualEsterelModule())
337                 .getEsterelTypeByID(((Element) anElementList.get(0))
338                     .getAttributeValue("id"));
339         this.myOP = this.getOperation(anElementList);
340         this.firstSignalExpression = this.getSignalExpression(
341             theparser,
342             anElementList,
343             xmldoc);
344     } else if (actualElement == "BinaryOp") {
345         this.isBinaryOp = true;
346         this.myType =
347             theparser
348                 .getEsterelModule(theparser.getActualEsterelModule())
349                 .getEsterelTypeByID(((Element) anElementList.get(0))
350                     .getAttributeValue("id"));
351         this.myOP = this.getOperation(anElementList);
352         this.firstSignalExpression = this.getSignalExpression(
353             theparser,
354             anElementList,
355             xmldoc);
356     }
357 }
358
359 /**
360  * secondSignalExpression = this.getSignalExpression(
361  * anElementList,
362  * xmldoc);
363  */
364
365 /**
366  * secondSignalExpression = this.getSignalExpression(
367  * anElementList,
368  * xmldoc),
369  * this.signalExpression = this.getSignalExpression(

```


C.2.21. EsterelSignal

Der Klassenaufbau ist in der Abbildung C.2.21 dargestellt.

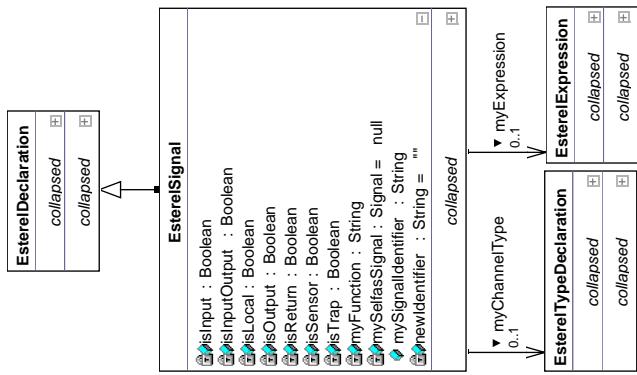


Abbildung C.16.: Klassendiagramm EsterelSignal

Auflistung C.26: Die Klasse EsterelSignal

```

import kiel.dataStructure.eventexp.IntegerSignal;
import kiel.dataStructure.eventexp.Signal;
import kiel.dataStructure.eventexp.StringSignal;
import kiel.fileInterface.estrel.EsterelParser;
import kiel.fileInterface.estrel.EsterelParserException;
import kiel.util.DOMHelpers;
import org.jdom.Element;

package kiel.fileInterface.estrel.estrel2estudio;
10 import java.util.List;
import java.util.Random;
import kiel.dataStructure.eventexp.CombineWithAdd;
import kiel.dataStructure.eventexp.CombineWithMult;
  
```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

private EsterelTypeDeclaration myChannelType; // child1
/* <p>
 * It represents estrel signals as global, local trap and
 * extern signals.
 */
20  * </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 0
 */
 /**
 * Is true if it is an output signal.
 */
 private boolean isOutput;
 */
 /**
 * 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).
 */
70  * 
 */
 * <p>
 * The expression of the signal. (Optional).
 */
 private EsterelExpression myExpression; //child1
 // 2
 /**
 * The function of the signal. (Optional).
 */
 private EsterelFunctionDeclaration myFunction; //child1
 /* 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 estrelSignal.
 * @return an integer.
 */
100  /**
 * the standard constructor.
 */
 public EsterelSignal() {
 super();
 }
 /**
 * @param theparser
 * a <code>EsterelParser</code>.
 * @param anElement
 * an <code>Element</code>.
 * @throws EsterelParserException
 * the exception.
 */
120  public EsterelSignal(final EsterelParser theparser,
 final Element anElement)
 throws EsterelParserException {
 this(
 theparser,
 "/*[@id=" + anElement.getAttributeValue("id")
 + ",]*");
 this.setName(anElement.getName());

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

240         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(anEstrelModule));
        } else {
            result = new IntegerSignal(
                this.getSignalIdentifier());
        }
    } else {
        if (this.myExpression != null) {
            result = new StringSignal(
                this.getSignalIdentifier(),
                this.myExpression
                    .convertToKielString(anEstrelModule));
        } else {
            result = new StringSignal(
                this.getSignalIdentifier());
        }
    }
    /*
     * throw new Estrel2EstudioException(
     * "EstrelSignal :" + "Unknown Type :"
     * + this.myChannelType.toString());
    */
    result = new Signal(
        this.getSignalIdentifier());
}
250
260
270
280
290
300
310
320
330
340
350
    /*
     * 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 anEstrelModule
     *          the estrel Module
     */
    public final void createSignalIdentifier(
        final EstrelModule anEstrelModule) {
        if (this.isTrap) {
            String preName = Estrel2EstudioProperties.getPreString();
            String postName = Estrel2EstudioProperties.getPostString();
            if (preName == null) {
                preName = "";
            }
            if (postName == null) {
                postName = "";
            }
            Random rand = new Random();
            this.mySignalIdentifier =
                postName +
                anEstrelModule
                    .getEstrelSignalByOriginalName(this.mySignalIdentifier)
                    != null) {
                    this.mySignalIdentifier =
                        preName +
                        this.mySignalIdentifier
                            + postName
                            + rand.nextInt();
                }
            }
        }
        /*
         * @return the type of the signal
         */
        public final EstrelTypeDeclaration getChannelType() {
            return this.myChannelType;
        }
        /*
         * @return the expression of the signal
         */
        public final EstrelExpression getExpression() {
            return this.myExpression;
        }
        /*
         * @return the name of the function
         */
        public final String getFunction() {
            return this.myFunction.getFunctionIdentifier();
        }
        /*
         * USE CAREFULLY.
         */
        /*
         * @return the originalSignalIdentifier.
         */
        public final String getOriginalSignalIdentifier() {
            return this.mySignalIdentifier;
        }
        /*
         * @return the signal identifier.
         */
        public final String getSignalIdentifier() {
            if (this.mySignalIdentifier == "") {
                return this.mySignalIdentifier;
            }
            return this.mySignalIdentifier;
        }
    }
}

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

        * @return is <code>true</code> if signal is an input signal.
        *
        * Public final boolean isInput() {
410             * @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 {
420            final int aTHREE = 3;
            final int aFOUR = 4;
            List anElementList = DOMHelpers.getElements(theXPath,
                theparser.getXMLDocument());
            if (anElementList.size() > 0) {
                this.setNames((Element) anElementList.get(0))
                    .setParentElement((Element) anElementList.get(0));
                this.setMLID((Element) anElementList.get(0))
                    .setParentElement((Element) anElementList.get(0));
                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)).getName());
430                    } else if (i == 1
                        && ((Element) anElementList.get(i)).getText() != null) {
                        this.myChannelType =
                            theparser
                                .getActualModule(theParser.getActualEsterelModule())
                                .getStreleModule().getByID(
                                    ((Element) anElementList.get(i)).getAttribute("id"));
                    }
                    this.myExpression = new EsterelExpression(
                        theparser,
                        .getActualModule(theParser.getActualEsterelModule())
                        .getStreleModule().getByID(
                            ((Element) anElementList.get(i)).getAttribute("id")));
440                } else if (i == 2
                    && ((Element) anElementList.get(i)).getName() != "NULL") {
                    this.myExpression = new EsterelExpression(
                        theXPath + "[local-name(.)=" +
                            ((Element) anElementList.get(i)).getName()
                            + "][" + ((Element) anElementList.get(i)).getAttribute("id"))
                        .getActualModule(theParser.getActualEsterelModule());
450                } 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")) {
460                    * Parses a signal. <br>
                     * Sideeffects: sets the class variables.
                     * @param theXPath the XPath which leads to the Expression
                     * @param theparser
                }
            }
        }
    }
}

```

```

        this.isLocal = true;
    } else if (aSignalType.equals("6")) {
        this.isTrap = true;
    } else {
        throw new EstrelParserException(
            "Unknown Signal Type: " + aSignalType);
    }
} else if (i == aFOUR
    && ((Element) anElementList.get(i)).getName() != "NULL") {
    this.myFunction =
        theparser
            .getEstrelModule(theparser.getActualEstrelModule())
            .getEstrelFunctionID(((Element) anElementList.get(i))
                .getAttributeValue("id"));
    /*new EstrelFunction(theparser,
        "/*[? [?=" +
        ((ELEMENT) anElementList.get(i))
            .getAttributeValue('id')
        + "]/*/",
        theparser.getXMLDocument());
    */
    if (i == 0) {
        throw new EstrelParserException(
            "No Signalname");
    } else {
        throw new EstrelParserException(
            "No SignalDeclaration found");
    }
}
/**/
/* @param aSignalIdentifier
   *          The mySignalIdentifier to set.
   */
public final void setMySignalIdentifier(
    * final String aSignalIdentifier) {
    this.mySignalIdentifier = aSignalIdentifier;
}

470
480
490
500
510
520
530
540
550
560
570
580
590
600
610
620
630
640
650
660
670
680
690
700
710
720
730
740
750
760
770
780
790
800
810
820
830
840
850
860
870
880
890
900
910
920
930
940
950
960
970
980
990
1000
1010
1020
1030
1040
1050
1060
1070
1080
1090
1100
1110
1120
1130
1140
1150
1160
1170
1180
1190
1200
1210
1220
1230
1240
1250
1260
1270
1280
1290
1300
1310
1320
1330
1340
1350
1360
1370
1380
1390
1400
1410
1420
1430
1440
1450
1460
1470
1480
1490
1500
1510
1520
1530
1540
1550
1560
1570
1580
1590
1600
1610
1620
1630
1640
1650
1660
1670
1680
1690
1700
1710
1720
1730
1740
1750
1760
1770
1780
1790
1800
1810
1820
1830
1840
1850
1860
1870
1880
1890
1900
1910
1920
1930
1940
1950
1960
1970
1980
1990
2000
2010
2020
2030
2040
2050
2060
2070
2080
2090
2100
2110
2120
2130
2140
2150
2160
2170
2180
2190
2200
2210
2220
2230
2240
2250
2260
2270
2280
2290
2300
2310
2320
2330
2340
2350
2360
2370
2380
2390
2400
2410
2420
2430
2440
2450
2460
2470
2480
2490
2500
2510
2520
2530
2540
2550
2560
2570
2580
2590
2600
2610
2620
2630
2640
2650
2660
2670
2680
2690
2700
2710
2720
2730
2740
2750
2760
2770
2780
2790
2800
2810
2820
2830
2840
2850
2860
2870
2880
2890
2900
2910
2920
2930
2940
2950
2960
2970
2980
2990
3000
3010
3020
3030
3040
3050
3060
3070
3080
3090
3100
3110
3120
3130
3140
3150
3160
3170
3180
3190
3200
3210
3220
3230
3240
3250
3260
3270
3280
3290
3300
3310
3320
3330
3340
3350
3360
3370
3380
3390
3400
3410
3420
3430
3440
3450
3460
3470
3480
3490
3500
3510
3520
3530
3540
3550
3560
3570
3580
3590
3600
3610
3620
3630
3640
3650
3660
3670
3680
3690
3700
3710
3720
3730
3740
3750
3760
3770
3780
3790
3800
3810
3820
3830
3840
3850
3860
3870
3880
3890
3900
3910
3920
3930
3940
3950
3960
3970
3980
3990
4000
4010
4020
4030
4040
4050
4060
4070
4080
4090
4100
4110
4120
4130
4140
4150
4160
4170
4180
4190
4200
4210
4220
4230
4240
4250
4260
4270
4280
4290
4300
4310
4320
4330
4340
4350
4360
4370
4380
4390
4400
4410
4420
4430
4440
4450
4460
4470
4480
4490
4500
4510
4520
4530
4540
4550
4560
4570
4580
4590
4600
4610
4620
4630
4640
4650
4660
4670
4680
4690
4700
4710
4720
4730
4740
4750
4760
4770
4780
4790
4800
4810
4820
4830
4840
4850
4860
4870
4880
4890
4900
4910
4920
4930
4940
4950
4960
4970
4980
4990
5000
5010
5020
5030
5040
5050
5060
5070
5080
5090
5091
5092
5093
5094
5095
5096
5097
5098
5099
5100
5101
5102
5103
5104
5105
5106
5107
5108
5109
5109
5110
5111
5112
5113
5114
5115
5116
5117
5118
5119
5119
5120
5121
5122
5123
5124
5125
5126
5127
5128
5129
5129
5130
5131
5132
5133
5134
5135
5136
5137
5138
5139
5139
5140
5141
5142
5143
5144
5145
5146
5147
5148
5149
5149
5150
5151
5152
5153
5154
5155
5156
5157
5158
5159
5159
5160
5161
5162
5163
5164
5165
5166
5167
5168
5169
5169
5170
5171
5172
5173
5174
5175
5176
5177
5178
5179
5179
5180
5181
5182
5183
5184
5185
5186
5187
5188
5189
5189
5190
5191
5192
5193
5194
5195
5196
5197
5198
5199
5199
5200
5201
5202
5203
5204
5205
5206
5207
5208
5209
5209
5210
5211
5212
5213
5214
5215
5216
5217
5218
5219
5219
5220
5221
5222
5223
5224
5225
5226
5227
5228
5229
5229
5230
5231
5232
5233
5234
5235
5236
5237
5238
5239
5239
5240
5241
5242
5243
5244
5245
5246
5247
5248
5249
5249
5250
5251
5252
5253
5254
5255
5256
5257
5258
5259
5259
5260
5261
5262
5263
5264
5265
5266
5267
5268
5269
5269
5270
5271
5272
5273
5274
5275
5276
5277
5278
5279
5279
5280
5281
5282
5283
5284
5285
5286
5287
5288
5289
5289
5290
5291
5292
5293
5294
5295
5296
5297
5298
5299
5299
5300
5301
5302
5303
5304
5305
5306
5307
5308
5309
5309
5310
5311
5312
5313
5314
5315
5316
5317
5318
5319
5319
5320
5321
5322
5323
5324
5325
5326
5327
5328
5329
5329
5330
5331
5332
5333
5334
5335
5336
5337
5338
5339
5339
5340
5341
5342
5343
5344
5345
5346
5347
5348
5349
5349
5350
5351
5352
5353
5354
5355
5356
5357
5358
5359
5359
5360
5361
5362
5363
5364
5365
5366
5367
5368
5369
5369
5370
5371
5372
5373
5374
5375
5376
5377
5378
5379
5379
5380
5381
5382
5383
5384
5385
5386
5387
5388
5389
5389
5390
5391
5392
5393
5394
5395
5396
5397
5398
5399
5399
5400
5401
5402
5403
5404
5405
5406
5407
5408
5409
5409
5410
5411
5412
5413
5414
5415
5416
5417
5418
5419
5419
5420
5421
5422
5423
5424
5425
5426
5427
5428
5429
5429
5430
5431
5432
5433
5434
5435
5436
5437
5438
5439
5439
5440
5441
5442
5443
5444
5445
5446
5447
5448
5449
5449
5450
5451
5452
5453
5454
5455
5456
5457
5458
5459
5459
5460
5461
5462
5463
5464
5465
5466
5467
5468
5469
5469
5470
5471
5472
5473
5474
5475
5476
5477
5478
5479
5479
5480
5481
5482
5483
5484
5485
5486
5487
5488
5489
5489
5490
5491
5492
5493
5494
5495
5496
5497
5498
5498
5499
5499
5500
5501
5502
5503
5504
5505
5506
5507
5508
5509
5509
5510
5511
5512
5513
5514
5515
5516
5517
5518
5519
5519
5520
5521
5522
5523
5524
5525
5526
5527
5528
5529
5529
5530
5531
5532
5533
5534
5535
5536
5537
5538
5539
5539
5540
5541
5542
5543
5544
5545
5546
5547
5548
5549
5549
5550
5551
5552
5553
5554
5555
5556
5557
5558
5559
5559
5560
5561
5562
5563
5564
5565
5566
5567
5568
5569
5569
5570
5571
5572
5573
5574
5575
5576
5577
5578
5579
5579
5580
5581
5582
5583
5584
5585
5586
5587
5588
5589
5589
5590
5591
5592
5593
5594
5595
5596
5597
5598
5598
5599
5599
5600
5601
5602
5603
5604
5605
5606
5607
5608
5609
5609
5610
5611
5612
5613
5614
5615
5616
5617
5618
5619
5619
5620
5621
5622
5623
5624
5625
5626
5627
5628
5629
5629
5630
5631
5632
5633
5634
5635
5636
5637
5638
5639
5639
5640
5641
5642
5643
5644
5645
5646
5647
5648
5649
5649
5650
5651
5652
5653
5654
5655
5656
5657
5658
5659
5659
5660
5661
5662
5663
5664
5665
5666
5667
5668
5669
5669
5670
5671
5672
5673
5674
5675
5676
5677
5678
5679
5679
5680
5681
5682
5683
5684
5685
5686
5687
5688
5689
5689
5690
5691
5692
5693
5694
5695
5696
5697
5698
5698
5699
5699
5700
5701
5702
5703
5704
5705
5706
5707
5708
5709
5709
5710
5711
5712
5713
5714
5715
5716
5717
5718
5719
5719
5720
5721
5722
5723
5724
5725
5726
5727
5728
5729
5729
5730
5731
5732
5733
5734
5735
5736
5737
5738
5739
5739
5740
5741
5742
5743
5744
5745
5746
5747
5748
5749
5749
5750
5751
5752
5753
5754
5755
5756
5757
5758
5759
5759
5760
5761
5762
5763
5764
5765
5766
5767
5768
5769
5769
5770
5771
5772
5773
5774
5775
5776
5777
5778
5779
5779
5780
5781
5782
5783
5784
5785
5786
5787
5788
5789
5789
5790
5791
5792
5793
5794
5795
5796
5797
5798
5798
5799
5799
5800
5801
5802
5803
5804
5805
5806
5807
5808
5809
5809
5810
5811
5812
5813
5814
5815
5816
5817
5818
5819
5819
5820
5821
5822
5823
5824
5825
5826
5827
5828
5829
5829
5830
5831
5832
5833
5834
5835
5836
5837
5838
5839
5839
5840
5841
5842
5843
5844
5845
5846
5847
5848
5849
5849
5850
5851
5852
5853
5854
5855
5856
5857
5858
5859
5859
5860
5861
5862
5863
5864
5865
5866
5867
5868
5869
5869
5870
5871
5872
5873
5874
5875
5876
5877
5878
5879
5879
5880
5881
5882
5883
5884
5885
5886
5887
5888
5889
5889
5890
5891
5892
5893
5894
5895
5896
5897
5898
5898
5899
5899
5900
5901
5902
5903
5904
5905
5906
5907
5908
5909
5909
5910
5911
5912
5913
5914
5915
5916
5917
5918
5919
5919
5920
5921
5922
5923
5924
5925
5926
5927
5928
5929
5929
5930
5931
5932
5933
5934
5935
5936
5937
5938
5939
5939
5940
5941
5942
5943
5944
5945
5946
5947
5948
5949
5949
5950
5951
5952
5953
5954
5955
5956
5957
5958
5959
5959
5960
5961
5962
5963
5964
5965
5966
5967
5968
5969
5969
5970
5971
5972
5973
5974
5975
5976
5977
5978
5979
5979
5980
5981
5982
5983
5984
5985
5986
5987
5988
5989
5989
5990
5991
5992
5993
5994
5995
5996
5997
5998
5998
5999
5999
6000
6001
6002
6003
6004
6005
6006
6007
6008
6009
6009
6010
6011
6012
6013
6014
6015
6016
6017
6018
6019
6019
6020
6021
6022
6023
6024
6025
6026
6027
6028
6029
6029
6030
6031
6032
6033
6034
6035
6036
6037
6038
6039
6039
6040
6041
6042
6043
6044
6045
6046
6047
6048
6049
6049
6050
6051
6052
6053
6054
6055
6056
6057
6058
6059
6059
6060
6061
6062
6063
6064
6065
6066
6067
6068
6069
6069
6070
6071
6072
6073
6074
6075
6076
6077
6078
6079
6079
6080
6081
6082
6083
6084
6085
6086
6087
6088
6089
6089
6090
6091
6092
6093
6094
6095
6096
6097
6098
6098
6099
6099
6100
6101
6102
6103
6104
6105
6106
6107
6108
6109
6109
6110
6111
6112
6113
6114
6115
6116
6117
6118
6119
6119
6120
6121
6122
6123
6124
6125
6126
6127
6128
6129
6129
6130
6131
6132
6133
6134
6135
6136
6137
6138
6139
6139
6140
6141
6142
6143
6144
6145
6146
6147
6148
6149
6149
6150
6151
6152
6153
6154
6155
6156
6157
6158
6159
6159
6160
6161
6162
6163
6164
6165
6166
6167
6168
6169
6169
6170
6171
6172
6173
6174
6175
6176
6177
6178
6179
6179
6180
6181
6182
6183
6184
6185
6186
6187
6188
6189
6189
6190
6191
6192
6193
6194
6195
6196
6197
6198
6198
6199
6199
6200
6201
6202
6203
6204
6205
6206
6207
6208
6209
6209
6210
6211
6212
6213
6214
6215
6216
6217
6218
6219
6219
6220
6221
6222
6223
6224
6225
6226
6227
6228
6229
6229
6230
6231
6232
6233
6234
6235
6236
6237
6238
6239
6239
6240
6241
6242
6243
6244
6245
6246
6247
6248
6249
6249
6250
6251
6252
6253
6254
6255
6256
6257
6258
6259
6259
6260
6261
6262
6263
6264
6265
6266
6267
6268
6269
6269
6270
6271
6272
6273
6274
6275
6276
6277
6278
6279
6279
6280
6281
6282
6283
6284
6285
6286
6287
6288
6289
6289
6290
6291
6292
6293
6294
6295
6296
6297
6298
6298
6299
6299
6300
6301
6302
6303
6304
6305
6306
6307
6308
6309
6309
6310
6311
6312
6313
6314
6315
6316
6317
6318
6319
6319
6320
6321
6322
6323
6324
6325
6326
6327
6328
6329
6329
6330
6331
6332
6333
6334
6335
6336
6337
6338
6339
6339
6340
6341
6342
6343
6344
6345
6346
6347
6348
6349
6349
6350
6351
6352
6353
6354
6355
6356
6357
6358
6359
6359
6360
6361
6362
6363
6364
6365
6366
6367
6368
6369
6369
6370
6371
6372
6373
6374
6375
6376
6377
6378
6379
6379
6380
6381
6382
6383
6384
6385
6386
6387
6388
6389
6389
6390
6391
6392
6393
6394
6395
6396
6397
6398
6398
6399
6399
6400
6401
6402
6403
6404
6405
6406
6407
6408
6409
6409
6410
6411
6412
6413
6414
6415
6416
6417
6418
6419
6419
6420
6421
6422
6423
6424
6425
6426
6427
6428
6429
6429
6430
6431
6432
6433
6434
6435
6436
6437
6438
6439
6439
6440
6441
6442
6443
6444
6445
6446
6447
6448
6449
6449
6450
6451
6452
6453
6454
6455
6456
6457
6458
6459
6459
6460
6461
6462
6463
6464
6465
6466
6467
6468
6469
6469
6470
6471
6472
6473
6474
6475
6476
6477
6478
6479
6479
6480
6481
6482
6483
6484
6485
6486
6487
6488
6489
6489
6490
6491
6492
6493
6494
6495
6496
6497
6498
6498
6499
6499
6500
6501
6502
6503
6504
6505
6506
6507
6508
6509
6509
6510
6511
6512
6513
6514
6515
6516
6517
6518
6519
6519
6520
6521
6522
6523
6524
6525
6526
6527
6528
6529
6529
6530
6531
6532
6533
6534
6535
6536
6537
6538
6539
6539
6540
6541
6542
6543
6544
6545
6546
6547
6548
6549
6549
6550
6551
6552
6553
6554
6555
6556
6557
6558
6559
6559
6560
6561
6562
6563
6564
6565
6566
6567
6568
6569
6569
6570
6571
6572
6573
6574
6575
6576
6577
6578
6579
6579
6580
6581
6582
6583
6584
6585
6586
6587
6588
6589
6589
6590
6591
6592
6593
6594
6595
6596
6597
6598
6598
6599
6599
6600
6601
6602
6603
6604
6605
6606
6607
6608
6609
6609
6610
6611
6612
6613
6614
6615
6616
6617
6618
6619
6619
6620
6621
6622
6623
6624
6625
6626
6627
6628
6629
6629
6630
6631
6632
6633
6634
6635
6636
6637
6638
66
```

C.2.22. Ester Statement

Der Klassenaufbau ist in der Abbildung C.2.22 dargestellt.

C. Java Code für die Transformation von Esterel in SyncCharts

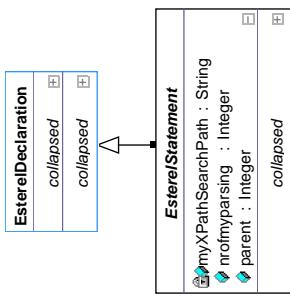


Abbildung C.17.: Klassendiagramm EsterelStatement

```

Aufflistung C.27: Die Klasse EsterelStatement

package kiel.fileInterface.estrel.estrel2estudio;
// import java.util.ArrayList;
import kiel.fileInterface.estrel.EsterelParser;
import kiel.fileInterface.estrel.EsterelParserException;

/*
 * <p> * This is the abstract superclass of all estrel statements
 * classes.
 * </p>
 * <p> * Copyright: Copyright (c) 2004
 * <p> * Company: Uni Kiel
 * </p>
 * <p> * Author: <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
 * </p>
 * <p> * Version Revision: 1.44 $ last modified $Date: 2006/02/06 18:35:29 $
 * </p>
 */
public abstract class EsterelStatement extends EsterelDeclaration {
    /**
     * A XPath search string which leads to this statement. */
    private String myXPathSearchPath;
    /**
     * Simple constructor.
     */
    public EsterelStatement() {
        this( null );
    }
    /**
     * Parses a Document and set the class variables.
     * It uses the <code>preParseEsterelStatement</code> methode
     * to parse the document.
     */
    public EsterelStatement(final EsterelParser theparser) {
        this.myXPathSearchPath = "";
        if (theparser != null) {
            theparser.setEsterelStatement(this);
            theparser.setEsterelInterface(esterel);
            theparser.setEsterelParser(theparser);
        }
    }
}

```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

* Returns a state representation of the estrel statement.          110
* @throws EsterelParserException
*   * is only delivered
*   * @see kiel.fileInterface.estrel.EsterelParserException
*   * @see kiel.fileInterface.estrel.EsterelParser
*/
public abstract void parseEsterelStatement(
    final EsterelParser theparser)
    throws EsterelParserException;

    /**
     * Parses an estrel statement in an <code>org.json.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
    /**
     * See #EsterelStatement(EsterelParser)
     * @see #parseEsterelStatement(EsterelParser)
     * @see kiel.fileInterface.estrel.EsterelParser
     * @see org.json.Document
     */
    public final void prepParseEsterelStatement(
        final EsterelParser theparser) {
        this.setPathname(((theparser.getElement()).getName()));
        this.setXMLID((theparser.getElement()).getAttributeValue("id"));
        if (theparser.getAida() > 0) {
            this.myXPathSearchPath = "//**"
                + "[local-name(.)=" + this.getName() + "][@id=" +
                + this.XMLID() + "]/**";
        } else {
            this.myXPathSearchPath = theparser.gettheElementXPathSearchString()
                + "[local-name(.)=" +
                + this.getFilename() +
                + "][@id=" +
                + this.XMLID() + "]/**";
        }
    }

    /**
     * Returns the name of an estrel statement. <br>
     * Sideffects: none
     */
    @return String name of the estrel statement
    public final String getEsterelStatementName() {
        return this.getfname();
    }

    /**
     * Returns the XPath string which leads in a
     * <code>org.json.Document</code> representation of an .exp file to
     * the estrel statement. <br>
     * Sideffects: none
     */
    @return String of the estrel statement
    @see org.json.Document
    public final String getXPathSearchString() {
        return this.myXPathSearchPath;
    }

    /**
     * Must be implemented in the subclasses. Parses an estrel statement
     * and set values. <br>
     * Sideffects: see at subclasses
     * @param theparser
     *   * an instance of a EsterelParser
     */
    public abstract void setSubStatements(
        final EsterelParser theparser)
        throws EsterelParserException;
*/

```

C. Java Code für die Transformation von Esterel in SyncCharts

```
    * Returns a string representation of an esterel statement.  
    *  
    * @param anEsterelModule  
    *      an esterel module representation  
    * @return a String representation of an esterel statement  
    *  
170     * @see kiel.fileInterface.estrel.estrel2estudio.EsterelModule  
     */  
    public abstract String toString(  
        final EsterelModule anEsterelModule);  
} // end of class EsterelStatement
```

C.2.23. Esterel Task Declaration

Der Klassenraum ist in der Abbildung C.2.23 dargestellt.

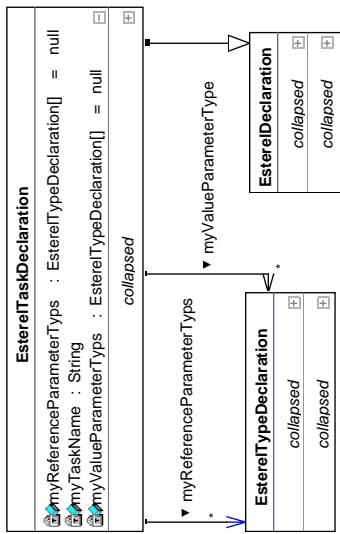


Abbildung C.18.: Klassendiagramm EsterelTaskDeclaration

```

Auflistung C.28: Die Klasse EsterelTaskDeclaration

package kiel.fileInterface.esterei.esterei2studio;

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

import kiel.fileInterface.esterei.EsterelParser;
import kiel.fileInterface.esterei.EsterelParserException;
import kiel.util.DOMHelpers;

import org.jdom.Element;

/*
 * <p>
 * It represents esterei Tasks .
 * </p>
 * <p>
 * Copyright: Copyright (c) 2005
 * </p>
 * <p>
 * Company: Uni Kiel
 * </p>
 */

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();
    }
}

```



```

160    public final String toString(
161        final EstrelModule anEstrelModule) {
162        String result = this.myTaskName
163        + " (";
164        if (this.myReferenceParameterTypes != null) {
165            for (int i = 0; i < this.myReferenceParameterTypes.length; i++) {
166                result += this.myReferenceParameterTypes[i].get TypeName()
167                + " ";
168            }
169        }
170        if (this.myValueParameterTypes != null) {
171            for (int i = 0; i < this.myValueParameterTypes.length; i++) {
172                result += this.myValueParameterTypes[i].get TypeName()
173                + " ";
174            }
175        }
176        result += " ) ( ";
177    }

```

C.2.24. EsterelTypeDeclaration

Der Klassenaufbau ist in der Abbildung C.2.24 dargestellt.

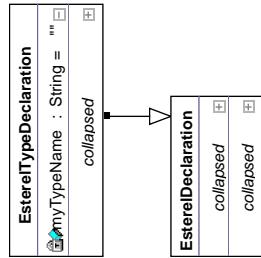


Abbildung C.19.: Klassediagramm EsterelTypeDeclaration

Auflistung C.29: Die Klasse EsterelTypeDeclaration

```

package kiel.fileInterface.esterel.esterel2estudio;

/*
 * <p>
 * Implements a type declaration.
 * </p>
 * 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 = "";
}

/*
 * @param aTypeName
 *          The myTypeName to set.
 */
  
```

The code defines the `EsterelTypeDeclaration` class, which extends the `EsterelDeclaration` class. It contains a private attribute `myTypeName` of type `String`, initialized to an empty string. The class includes Javadoc comments for its methods and fields, providing information about its implementation and usage.

```
/*
 * @return the string representation.
 */
public String toString() {
    return this.myTypeName;
}

/**
 * Set the type name.
 */
public void setTypeName(final String aTypeName) {
    final String myTypeName = aTypeName;
    this.myTypeName = myTypeName;
}
```

C.2.25. Esterel Variable

Der Klassenaufbau ist in der Abbildung C.2.25 dargestellt.

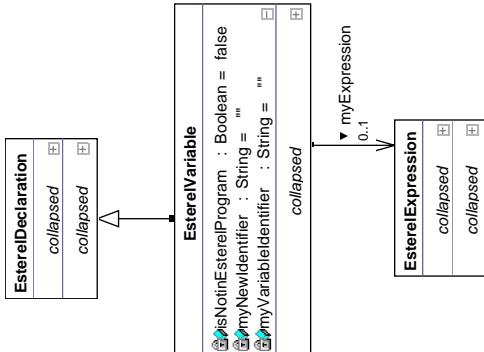


Abbildung C.20.: Klassediagramm EsterelVariable

```

package kiel.fileInterface.estrel.estrel2estudio;
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.estrel.EsterelParser;
import kiel.fileInterface.estrel.EsterelParserException;
import kiel.util.DOMHelpers;

// import org.jdom.Element;
import org.jdom.Element;

/*
 * <p>
 * It represents local estrel variables .
 */
20  * <p>
 * Copyright : Copyright (c) 2005
 * </p>
 * <p>
 * Company : Uni Kiel
 * </p>
 * <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>
 */
30

```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

/*
 * @param theparser
 * extends EsterelDeclaration {
90   * @param the estrel parser
   * @param theXPath the XPath to the EsterelVariable
   * @throws EstrelParserException
   *         if something goes wrong at the parsing.
   */
public EsterelVariable(final EstrelParser theparser,
final String theXPath)
throws EstrelParserException {
super();
this.myVariableIdentifier = null;
this.myExpression = null;
this.myChannelType = null;
if (theparser != null) {
parseEsterelStatement(theparser,
theXPath);
}
}

/**
 * If it is a new variable.
 */
private boolean isNotInEstrelProgram = false;
40   * The type of the signal. (Optional).
   */
private EsterelTypeDeclaration myChannelType; // child1
   */
private EsterelTypeDeclaration myNewIdentifier = ""; // child0
// 2
50   * The new identifier name.
   */
private String myNewIdentifier = ""; // child 0
   */
* Me as Variable.
private Variable mySelfasVariable = null;
   */
private String myVariableIdentifier = ""; // child 0
   */
* The name of the signal.
60   * The name of the signal.
   */
private String myVariableIdentifier = ""; // child 0
   */
* the standard constructor.
   */
public EsterelVariable() {
super();
}
}

70   * @param theparser a<code> EstrelParser</code>
   * @param anElement an Element
   * @throws EstrelParserException the Exception
   */
public EsterelVariable(final EstrelParser theparser,
final Element anElement)
throws EstrelParserException {
this(
  theparser, /*@ [id=,*/
  this.setName(anElement.getAttributeValue("id") + "]/*");
  this.setNamespace(anElement.getNamespaceURI());
  this.setXMLID(anElement.getAttributeValue("id"));
}
}

80   */
   * Creates an EsterelVariable and fills the class variables. <br>
   * Sideeffects: class variables are changed.
}

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

    } else {
        result = new DoubleVariable(
            this.myVariableIdentifier);
    }
}
else {
    if (this.myExpression != null) {
        result = new IntegerVariable(
            this.myVariableIdentifier,
            this.myExpression.convertToKiel(
                anEsterelModule,
                theLocalEvents,
                theLocalVariables));
    }
    else {
        result = new IntegerVariable(
            this.myVariableIdentifier);
    }
}
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 support 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 support for double:
                             *this.myExpression
                             *.convertToKiel(anEsterelModule));
                */
            }
        }
    }
}
}
else {
    if (this.myExpression != null) {
        result = new StringVariable(
            this.getVariableIdentifier(),
            null,
            this.myChannelType.getTypeName());
    }
}
/* 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.
 */
public final void createNewIdentifierName(
    final EsterelModule anEsterelModule) {
    if (this.isInInEsterelProgram) {
        String preName = Esterel2EsterelProperties.getPreString();
        String postName = Esterel2EsterelProperties.getPostString();
        if (preName == null) {
            preName = "";
        }
        if (postName == null) {
            postName = "";
        }
        Random rand = new Random();
        this.myNewIdentifier = preName + this.myVariableIdentifier
            + postName;
        while (anEsterelModule
            .getEsterelVariableByName(this.myNewIdentifier)

```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

!= null) {
    this.myNewIdentifier = preName + this.myVariableIdentifier
        + PostName
        + rand.nextInt();
}

}

/**
 * @return Returns the myChannelType.
 */
public final EstrelTypeDeclaration getChannelType() {
    return this.myChannelType;
}

/**
 * @return Returns the myExpression.
 */
public final EstrelExpression getExpression() {
    return this.myExpression;
}

/**
 * USE CAREFULLY.
 *
 * @return the originalVariableIdentifier.
 */
public final String getOriginalVariableIdentifier() {
    return this.myVariableIdentifier;
}

/**
 * @return the variable identifier.
 */
public final String getVariableIdentifier() {
    if (this.myNewIdentifier == "") {
        return this.myVariableIdentifier;
    }
    return this.myNewIdentifier;
}

/**
 * Parses a variable. <b>P</b>
 * Sideeffects: sets the class variables.
 *
 * @param theXPath the XPath which leads to the Expression
 * @param theparser the estrel parser
 * @throws EstrelParserException
 * @see kiel.fileInterface.estrel.EstrelParserException
 */
public final void parseEstrelStatement(
    final EstrelParser theparser,
    final String theXPath)
throws EstrelParserException {
    List anElementList = DOMHelpers.getElements(theXPath,
        theparser.getDocument());
    if (anElementList.size() > 0
        && ((Element) anElementList.get(0)).getParentElement()
            .getTagName())
        .compareTo("VariableSymbol") != 0) {
        throw new EstrelParserException(
            "EstrelVariable: 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.getActualEstrelModule()
                    .getEstrelModule(theparser.getActualEstrelModule())
                        .getEstrelTypeByID(
                            ((Element) anElementList.get(i))
                                .getAttributeValue("id"));
        } else if (i == 2
            && ((Element) anElementList.get(i)).getName() != "NULL") {
            this.myExpression = new EstrelExpression(
                theparser,
                theXPath
                    + "[local-name(.) ="
                    + ((Element) anElementList.get(i)).getName()
                    + "] [id='"
                    + ((Element) anElementList.get(i))
                        .getAttributeValue("id"))
                    + "']");
        } else if (i == 0) {
            throw new EstrelParserException(
                "EstrelVariable: "
                + "No Variablename");
        }
    }
}

/**
 * @param aChannelType
 * @param the ChannelType to set.
 */
public final void setChannelType(
    final EstrelTypeDeclaration aChannelType) {
    this.myChannelType = aChannelType;
}

/**
 * @param aExpression
 * @param the Expression to set.
 */
public final void setExpression(

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

230
    final EsterelExpression aExpression {
        this.myExpression = aExpression;
    }
    /**
     * @param aVariableIdentifier
     *         The myVariableIdentifier to set.
     */
    public final void setVariableIdentifier(
        final String aVariableIdentifier) {
        this.myVariableIdentifier = aVariableIdentifier;
    }
    /**
     * Creates 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) {
370
            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;
    }
}
410

```

C.2.26. EveryEstrelStatement

Der Klassenaufbau ist in der Abbildung C.2.26 dargestellt.

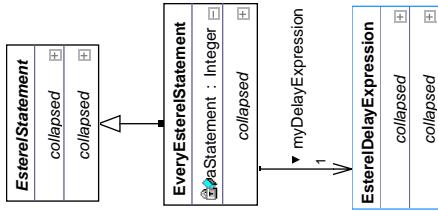


Abbildung C.21.: Klassendiagramm EveryEstrelStatement

Auflistung C.31: Die Klasse EveryEstrelStatement

```

20   * <p> This is a subclass of <code>EstrelStatement</code>
21   * classes implements the every statement. </p>
22   * <p>Copyright: Copyright (c) 2005</p>
23   * <p>Company: Uni Kiel</p>
24   *
25   * @author <a href="mailto:lkw@informatik.uni-kiel.de">Lars Kuehl</a>
26   * @version Revision: 1.32 $ last modified $Date: 2006/02/17 20:00:22 $
27   */
28 public class EveryEstrelStatement extends EstrelStatement {
29     /**
30      * The index number of the substatement in an
31      * <code>EstrelModule</code>
32      * @see kiel.fileInterface.estrel.estrel2estudio.EstrelModule
33      */
34     private int aStatement;
35     /**
36      * The delay expression as <code>EstrelDelayExpression</code>
37      * @see EstrelDelayExpression
38      */
39
40 import kiel.util.DOMHelpers;
41 import kiel.util.StateCharHelpers;
42 import org.json.Element;
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
287
288
289
289
290
291
292
293
294
295
296
297
297
298
299
299
300
301
302
303
304
305
305
306
307
307
308
309
309
310
311
312
312
313
313
314
314
315
315
316
316
317
317
318
318
319
319
320
320
321
321
322
322
323
323
324
324
325
325
326
326
327
327
328
328
329
329
330
330
331
331
332
332
333
333
334
334
335
335
336
336
337
337
338
338
339
339
340
340
341
341
342
342
343
343
344
344
345
345
346
346
347
347
348
348
349
349
350
350
351
351
352
352
353
353
354
354
355
355
356
356
357
357
358
358
359
359
360
360
361
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
```

C. Java Code für die Transformation von Esterel in SyncCharts

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

* Parses an estrel statement in an <code>org.jdom.Document</code>
* representation of an .exp file
* if you have not called the constructor
* <code> EstrelParser</code>
* use the methode <code>preParseEstrelStatement</code>
* before <code>parseEstrelStatement</code>
* <br>Sideeffects:
* the class variables are set
* Changes <code>EstrelParser</code>>class variables
* @throws EstrelParserException is only delivered
* @see kiel.fileInterface.estrel.EstrelParser
* @see EstrelStatement#parseEstrelStatement(EstrelParser)
* @see EstrelStatement#parseEstrelStatement(EstrelParser)
* @see kiel.fileInterface.estrel.EstrelParser
* @see org.jdom.Document
* Fills the substatements in the estrel module.
150    * In this class it is fills the do Statements if there are any
    *<br> Sideeffects: none
    * @param theparser the actual parser
    * @throws EstrelParserException is only delivered
    * @see kiel.fileInterface.estrel.EstrelParser
    */
    public final void setSubStatements(final EstrelParser theparser)
        throws EstrelParserException {
        int i = 0;
        List anElementList = null;
        anElementList =
        DOMHelpers.getElements(
            theparser.getXMLDocument(),
            this.getXPathSearchString());
        if (anElementList.size() == 2) {
            this.astatement = theparser.getAddAt();
            theparser.getEstrelModule(
                theparser.getActualEstrelModule()
                .addStatementToModule(
                    theparser.getAddAt(),
                    theparser.createStatement((Element) (anElementList.get(i)))));
        }
    }

    /**
     * Implements the abstract methode from <code>EstrelStatement</code>.
     * Returns a string representation of an estrel statement.
     * @param anEstrelModule an estrel module representation
     * @return a String representation of an estrel statement
     * @see kiel.fileInterface.estrel2estudio.EstrelStatement
     */
    public final String toString(final EstrelModule anEstrelModule)
        throws this.getEstrelStatementName()
        + " "
        + this.myDelayExpression.toString(anEstrelModule)
        + " "
        + this.myDelayExpression.toSring(anEstrelModule)
        + " "
        + " " do "\n"
        + (
            (EstrelStatement) anEstrelModule.getEstrelProgram().get(
                this.astatement).toString(
                    anEstrelModule))
        + "end "
        + this.getEstrelStatementName();
    }

    /**
     * Fills the substatements in the estrel module.
160    public final void parseEstrelStatement(final EstrelParser theparser)
        throws EstrelParserException {
        final int index = 1;
        List anElementList = null;
        anElementList =
        DOMHelpers.getElements(
            theparser.getXMLDocument(),
            this.getXPathSearchString());
        if (anElementList.size() == 2) {
            theparser.incStatementCounter();
            this.myDelayExpression =
            new EstrelDelayExpression(
                theparser,
                "/*local-name(.)=" +
                ((Element) anElementList.get(index)).getName() +
                "*/");
            ((Element) anElementList.get(index))
                .setAttributeValue('id')
                + "] [id=",
            theparser.getXMLDocument());
        } else {
            throw new EstrelParserException(
                this.getEstrelStatementName() + " statement wrong",
                theparser);
        }
        //else if size
170    }
    */
180    /**
     * Fills the substatements in the estrel module.
190    */
200    /**
     * In this class it is fills the do Statements if there are any
     *<br> Sideeffects: none
     * @param theparser the actual parser
     * @throws EstrelParserException is only delivered
     * @see kiel.fileInterface.estrel.EstrelParser
     */

```

C.2.27. ExitEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.27 dargestellt.

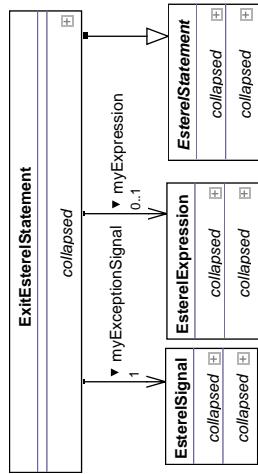


Abbildung C.22.: Klassendiagramm ExitEsterelStatement

Auflistung C.32: Die Klasse ExitEsterelStatement

```

package kiel.fileInterface.esterel.esterel2estudio;
/*
 * Copyright : Copyright (c) 2004
 *             * Company : Uni Kiel
 *             * 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;
    /**
     * Simple constructor.
     */
    public ExitEsterelStatement() {
        super();
    }
}

/*
 * This is a subclass of <code>EsterelStatement</code>
 * Classes implements the exit statement.
 */
/** Parses a Document and set the class variables.
 * It calls also the super constructor <code>ExitEsterelStatement
 */

```

```

String myName = this.myExceptionSignal.getSignalIdentifier();
String emithese = myName;
if (theTreeTrapSignals.get(i) instanceof TrapEstrelStatement) {
    emithese += "(" + this.myExpression.toString(anEstrelModule)
        + ")";
}
for (int i = theTreeTrapSignals.size() - 1; i > 0 && !found; i--) {
    if (theTreeTrapSignals.get(i) instanceof TrapEstrelStatement) {
        TrapEstrelStatement aTrap =
            (TrapEstrelStatement) theTreeTrapSignals.get(i);
        EstrelSignal[] allExceptionDeclaration =
            aTrap.getAllExceptionDeclarationList();
        for (int j = allExceptionDeclaration.length - 1; j > -1; j--) {
            if (allExceptionDeclaration[j].getSignalIdentifier()
                .compareTo(myName) == 0) {
                found = true;
            } else {
                emithese += ", " + aTrap.getMyHaltSignal()
                    .getSignalIdentifier();
            }
        }
    } else {
        AbortEstrelStatement abort =
            (AbortEstrelStatement) theTreeTrapSignals.get(i);
        emithese += ", " + abort.getMyHaltSignal()
            .getSignalIdentifier();
    }
} //for
TransitionLabel theLabelOfATransition = new CompoundLabel();
Event anEvent = this.myExceptionSignal.convertToKiel(anEstrelModule);
emithese += ", " + anEvent.getMyHaltSignal()
    .getSignalIdentifier();
}
theLabelOfATransition =
    .setEffect(StateChartHelpers
        .getActions(anEstrelModule.getEstrelStateChart(),
        theLocalVariables,
        theLocalEvents,
        emithese));
if (anEvent != null) {
    ((CompoundLabel) theLabelOfATransition)
        .createFinalOr(this.theLabelOfATransition).getEffect() == null) {
        theLabelOfATransition = new StringLabel(
            emithese);
    }
}
initialArc.setLabel(theLabelOfATransition);
if (isFinalState) {
    return StateChartHelpers
        .createFinalOr(this.getEstrelStatementName()
            + "state",
        new Node[] {
            anInitialState, theEndState });
}
return StateChartHelpers
    .createOr(this.getEstrelStatementName()
        .createNode("state",
        new Node[] {
            anInitialState, theEndState });
}
//public final kiel.dataStructure.State convertToKiel(

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

    * Fills the substatements in the esterel module in this class it is
    * empty <br>
    * Sideeffects: none
    *
    * Implements the abstract methode from <code>EsterelStatement</code>.
    * Parses an esterel statement in an <code>org.json.Document</code>
    * representation of an exp file, if you have not called the
    * constructor <code> EsterelParser</code> theparser before
    * 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 EsterelStatementEsterelStatement(EsterelParser)
    * @see EsterelStatementParseEsterelStatement(EsterelParser)
    * @see EsterelStatementPreParseEsterelStatement(EsterelParser)
    * @see kiel.fileinterface.esterel.EsterelParser
    * @see org.json.Document
    */
    public final void parseEsterelStatement(
        final EsterelParser theparser
    ) throws EsterelParserException {
        final int toomuch = 3;
        List anElementList = null;
        DOMHelpers getElements(theparser.getXMLDocument(),
            this.getXPathSearchString());
        if (anElementList.size() > 0 && anElementList.size() < toomuch) {
            this.myExceptionSignal =
                theparser
                    .getEsterelModule(theparser.getactualEsterelModule())
                    .getEsterelSignalByID(
                        this.getSignal(anElementList));
            if (anElementList.size() > 1 && anElementList.size() < toomuch) {
                final int theIndex = 1;
                this.myExpression = new EsterelExpression(
                    theparser
                        .getXPathSearchString() + "[local-name(),"
                        + ((Element) anElementList.get(theIndex))
                        .getName()
                        + ",]" + "[@id=" +
                        ((Element) anElementList.get(theIndex))
                        .getAttributeValue("id")
                        + "]/");
            } else {
                throw new EsterelParserException(
                    this.getEsterelStatementName() + " no signal expression",
                    theparser);
            }
        } //if >0
    } // end of class
}

```

C.2.28. HaltEstrelStatement

Der Klassenaufbau ist in der Abbildung C.2.28 dargestellt.

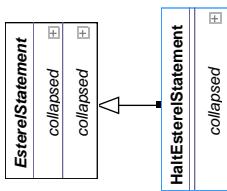


Abbildung C.23.: Klassendiagramm HaltEstrelStatement

Auflistung C.33: Die Klasse HaltEstrelStatement

```

package kiel.fileInterface.estrel.estrel2estudio;
import java.util.ArrayList;
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.Node;
import kiel.dataStructure.SimpleState;
import kiel.dataStructure.StatefulEstrelParser;
import kiel.fileInterface.estrel.EstrelParserException;
import kiel.util.StatechartHelpers;
import kiel.util.StatechartParser;
/*
 * <p> This is a subclass of <code>EstrelStatement</code>
 * classes implements the nothing statement.</p>
 */
10   /**
 * <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: HaltEstrelStatement.java,v $
 * Revision 1.37 2006/02/06 18:35:29 lku
 * *** empty log message ***
 * Revision 1.36 2006/11/10 12:26:26 lku
 * *** empty log message ***
 * Revision 1.35 2006/11/10 12:26:26 lku
 * *** empty log message ***
 * Revision 1.27 2006/10/04 11:33:29 lku
 * *** empty log message ***
 * Revision 1.26 2006/10/04 11:33:29 lku
 * *** empty log message ***
 * Revision 1.25 2006/10/04 11:33:29 lku
 * *** empty log message ***
 * Revision 1.24 2006/10/04 11:33:29 lku
 * *** empty log message ***
 * Revision 1.23 2006/10/04 11:33:29 lku
 * *** empty log message ***
 * Revision 1.22 2006/10/04 11:33:29 lku
 * *** empty log message ***
 * Revision 1.21 2006/10/04 11:33:29 lku
 * *** empty log message ***
 * Revision 1.20 2006/10/04 11:33:29 lku
 * *** empty log message ***
 * Revision 1.19 2006/10/04 11:33:29 lku
 * *** empty log message ***
 * Revision 1.18 2006/10/04 11:33:29 lku
 * *** empty log message ***
 * Revision 1.17 2006/05/12 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
 * add Javadoc + check for project manual code conventions
 * Revision 1.8 2005/01/14 15:00:00 lku
 * <br>
 * fix Javadoc
 */
public class HaltEstrelStatement extends EstrelStatement {
    /**
     * Simple constructor.
     */
    public HaltEstrelStatement() {
        super();
    }
}
  
```

C. Java Code für die Transformation von Esterel in SyncCharts

C.2.29. IfEstrelStatement

Der Klassenaufbau ist in der Abbildung C.2.29 dargestellt.

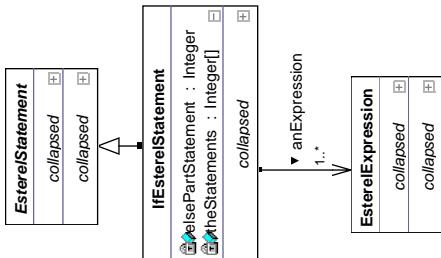


Abbildung C.24.: Klassendiagramm IfEstrelStatement

Auflistung C.34: Die Klasse IfEstrelStatement

```

20   * <p>This is a subclass of <code>EsterelStatement</code>
21   * classes implements the if statement /<p>
22   * <p>Copyright: Copyright (c) 2004</p>
23   * <p>Company: Uni Kiel</p>
24   * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
25   * @version Revision: 1.38 $ last modified $Date: 2006/02/06 18:35:29 $
26   */
27   public class IfEstrelStatement extends EsterelStatement {
28       /* The SignalExpressions */
29       private EsterelExpression[] anExpression;
30       /** the index number of the else part substatement in an
31       * <code>EsterelModule</code>
32       * @see kiel.fileInterface.estrel.estrel2estudio.EsterelModule
33       */
34       private int elsePartStatement;
35       /** the index numbers of the then part substatements in an
36       * <code>EsterelModule</code>
37       * @see kiel.fileInterface.estrel.estrel2estudio.EsterelModule
38       */
39   }
  
```

C. Java Code für die Transformation von Esterel in SyncCharts

```

/*
 * private int [] theStatements;
 */
40  /**
 * the standard constructor.
 */
41  public IfEsterelStatement() {
42      super();
43      this.theStatements = null;
44      this.elsePartStatement = -1;
45      this.anExpression = null;
46  }
47  /**
 * Parses a Document and set the class variables.
 */
50  * <br>It calls also the super constructor</code>EsterelStatement
51  * (EsterelParser theparser)</code> .
52  * <br>Calls the <code>parseEsterelStatement</code> method
53  * <br>>Sideeffects:
54  * Changes <code>EsterelParser</code>>class variables
55  * @param theparser an instance of a EsterelParser
56  * @throws EsterelParserException is only delivered
57  * @see kiel.fileInterface.estrel.EsterelParserException
58  * @see #parseEsterelStatement(EsterelParser)
59  * @see EsterelStatement#EsterelStatement(EsterelParser)
60  * @see kiel.fileInterface.estrel.EsterelParser
61  */
62  public IfEsterelStatement(final EsterelParser theparser)
63      throws EsterelParserException {
64      super(theparser);
65      this.theStatements = null;
66      this.elsePartStatement = -1;
67      this.anExpression = null;
68      if (theparser != null) {
69          this.parseEsterelStatement(theparser);
70      }
71  }
72  /**
 * Returns a state representation of the estrel statement.
 */
73  * @param anEsterelModule the estrel module which
74  * becomes a KIEL Statechart
75  * @param isFinalState is true if the new state has to be
76  * a final state
77  * @param theLocalEvents Stack with the local events
78  * @param theLocalVariables Stack with all local variables
79  * @param theTreeTrapSignals
80  * @throws Esterel2StudioException
81  * if the conversion to Kiel is not working. <br>
82  * Sideeffects: none
83  * @return a State
84  * @see kiel.fileInterface.estrel.estrel2estudio.EsterelModule
85  * @see kiel.datastructure.State
86  */
87  public final kiel.datastructure.State convertToKiel(
88      final EsterelModule anEsterelModule,
89      final boolean isFinalState,
90      final ArrayList<theLocalEvents,
91      theLocalVariables,
92      theTreeTrapSignals>;

```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

    } else {
        theEndStates[nrOfEndStates - 1] = new FinalSimpleState();
    }
    ConditionalTransition aChoiceTransition = new ConditionalTransition();
    StatechartHelpers.set(
        aChoiceTransition,
        aPresentTest,
        nrOfEndStates,
        theEndStates,
        theInitialArc.setSource(nrOfEndStates - 1));
    theInitialArc.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;
    if (isFinalState) {
        return StatechartHelpers.createFinalOR(
            this.getEstrelStatementName() + "state",
            theNodes);
    }
    return StatechartHelpers.createOR(
        this.getEstrelStatementName() + "state",
        theNodes);
}
/***
 * Implements the abstract methode from <code>EstrelStatement</code>
 * Parses an estrel statement in an <code>org.jdom.Document</code>
 * representation of an .exp file
 * if you have not called the constructor
 * <code> EstrelStatement(EstrelParser theparser)</code>
 * use the methode <code>preParseEstrelStatement</code>
 * before <code>parseEstrelStatement</code>
 * <br>Sideeffects:
 * the class variables are set
 * Changes <code>EstrelParser</code> class variables
 * @param theparser an instance of <code>EstrelParser</code>
 * @throws EstrelParseException estrel.ParserException
 * @see kiel.fileInterface.estrel.EstrelParser
 * @see EstrelStatement#parseEstrelStatement(EstrelParser)
 * @see EstrelStatement#parseEstrelStatement(EstrelParser)
 * @see kiel.fileInterface.estrel.EstrelParser
 * @see org.jdom.Document
 */
public final void parseEstrelStatement(final EstrelParser theparser)
    throws EstrelParserException {
    List anElementList =
        DOMHelpers.getElements(
            theparser.getDocument(),
            this.getxpathSearchString());
    int numberofPredicated = 0;
    if (((Element) anElementList.get(anElementList.size() - 1)).getName()
        == "PredicatedStatement") {
        numberofPredicated = anElementList.size() - 1;
    }
    ConditionalTransition aChoiceTransition =
        new ConditionalTransition();
    StatechartHelpers.set(
        aChoiceTransition,
        aPresentTest,
        nrOfEndStates,
        theEndStates,
        theInitialArc.setSource(nrOfEndStates - 1));
    theInitialArc.setTarget(aPresentTest);
    Node[] theNodes = new Node[theEndStates.length + 2];
    for (int i = 0; i < numberofPredicated; i++) {
        this.anExpression[i] = null;
        this.theStatements[i] = null;
    }
    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)).getAttribute(
                    "id"));
            List aPredicatedElementList =
                DOMHelpers.getElements(
                    theparser.getDocument(),
                    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)).getAttribute(
                    "id"));
            this.anExpression[theCaseIndex] =
                new EstrelExpression(
                    theparser,
                    aSubPredicatedStatementXPathSearchString,
                    theparser.getDocument());
        } else {
            throw new EstrelParserException(
                this.getEstrelStatementName() + " wrong statement",
                theparser);
        }
    }
}

```

C. Java Code für die Transformation von Esterel in SyncCharts

C.2.30. LocalSignalDeclarationEsterelStatement

Auffistung C.35: Die Klasse LocalSignalDeclarationEsterei Statement benennt ein Intace of a EsterelParser

C. Java Code für die Transformation von Esterel in SyncCharts

```

        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(
                true,
                theLocalVariables,
                theLocalEvents,
                theLocalSignals);
        /*true, because we have only one state
        // we added all the localsignals to theLocalEvents
        // 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);
        aStatementState.addChildNode(aStatementState);
        aState.addChildNode(anInitialState);
        return aState;
    } //public final kiel.dataStructure.State convertToKiel(
    /**
     * Implements the abstract methode from <code>EsterelStatement</code>.
     * Parses an esterel statement in an <code>org.jdom.Document</code>
     * representation of an .exp file, if you have not called the
     * constructor <code> EsterelParser esterelParser theparser</code>
     * use the methode <code>parseEsterelStatement</code>><br>
     * <<code>parseEsterelStatement</code>><br>
     * Sideffects: the class variables are set Changes
     * <code>EsterelParser</code> class variables
     * @param theparser
     * an instance of <code>EsterelParser</code>
     * @throws EsterelParserException
     * @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 {
        String theSymbolTable = "";
        ArrayList aSignalDeclarationList = new ArrayList();
        List anElementList = DOMHelpers.getElements(theparser.getDocument());
        DOMHelpers.getElements(theparser.getDocument());
        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.get XPathSearchString()
            + "[Local-name(.)='"
            + ((Element) anElementList.get(1)).getName()
            + "']/*";
        }
        else if (((Element) anElementList.get(1)).getName()
            .compareTo("Ref") == 0) {
            theSymbolTable = "/*[@id="
            + ((Element) anElementList.get(1))
            .getAttribute("id")
            + ",*/]/*";
        }
        else if (((Element) anElementList.get(1)).getName()
            .compareTo("Element") == 0) {
            theSymbolTable = "/*[@id="
            + ((Element) anElementList.get(1))
            .getAttribute("id")
            + ",*/]/*";
        }
        else {
            throw new EsterelParserException(
                this.getEsterelStatementName()
                + " statements incorrect",
                theparser);
        }
        List theSymbolTableElementList =
        DOMHelpers.getElements(theparser.getDocument(),
            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.getAttribute("id").compareTo("Ref") == 0) {
                // search for the ref
                aSignalDeclarationList.add(theparser
                    .getEsterelModule(theparser.getActualEsterelModule())
                    .getEsterelSignalByID(aSignalElement
                        .getAttributeValue("id")));
            }
            else if (aSignalElement.getAttribute("id").compareTo("Signal") == 0) {
                theparser.add(theparser
                    .getEsterelModule(
                        theparser.getActualEsterelModule()
                        .getEsterelSignalByName()
                        .getAttributeValue("id")));
            }
        }
        else {
            throw new EsterelParserException(
                this.getEsterelStatementName()
                + " statements incorrect",
                theparser);
        }
    }
    if (aSignalDeclarationList

```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

C.2.31. LocalVariableEsterelStatement

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

theLocalEvents,
theLocalVariables);
}
stateCharHelpers.addLocalVars(aState, theVariables);
int oldsize = theLocalVariables.size();
theLocalVariables.addAll(aState.getVariables());
State aStatementState =
(
    (EstrelStatement) anEstrelModule.getEstrelProgram().get(
        this.aStatement).convertToKiel(
            anEstrelModule,
            true,
            theLocalEvents,
            theLocalVariables,
            theTreeTrapSignals);
    /*true, because we have only one state
    // we added all the localVariables to theLocalEvents
    // 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);
    aStatementState.addChildNode(aState);
    aState.addChildNode(anInitialState);
    return aState;
} //public final kiel.dataStructure.State convertToKiel(
/***
 *Implements the abstract methode from <code>EstrelStatement</code>.
 * Parses an estrel statement in an <code>org.jdom.Document</code>
 * representation of an .exp file.
 * If you have not called the constructor
 * <code> EstrelStatement(EstrelParser theparser)</code>
 * use the methode <code>preparseEstrelStatement</code>
 * before <code>parseEstrelStatement</code>
 */
* <br>Sideeffects:
 * the class variables are set
 * Changes <code>EstrelParser</code>-class variables
 * @param theparser an instance of <code>EstrelParser</code>
 * Throws EstrelParseException is only delivered
 * @see kiel.fileInterface.estrel.EstrelParserException
 * @see EstrelStatement#EstrelStatement(EstrelParser theparser)
 * @see #parseEstrelStatement(EstrelParser)
 * @see EstrelStatement#parseEstrelStatement(EstrelParser)
 */
* @since 1.7
 */
public final void parseEstrelStatement(final EstrelParser theparser)
throws EstrelParserException {
    ArrayList<VariableDeclaration> = new ArrayList();
    String theSymbolTable = "";
    List anElementList =
DOMHelpers.getElements(
    theparser.getXMLDocument(),
    this.get XPathSearchString(),
    //getAll 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))
            .getTagName()
            .compareTo("SymbolTable") ==
0) {
            theSymbolTable =
this.get XPathSearchString()
+ "[Local-name(.)='"
+ ((Element) anElementList.get(1)).getTagName()
+ "']");
        }
    }
    else if (
        ((Element) anElementList.get(1)).getTagName() .compareTo("Ref")
        == 0) {
            theSymbolTable =
"/*[@id='"
+ ((Element) anElementList.get(1)).getTagName()
+ "']");
        }
    }
    else {
        throw new EstrelParserException(
            this.getEstrelStatementName() + " 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.getAttribute("id") == 0) {
        // search for the refenc ref
        aVariableDeclarationList.add(
            new EstrelVariable(
                theparser,
                "/*[@id='"
                + aVariableElement.getAttribute("id")
                + "']");
    }
    else if (
        aVariableElement.getTagName() .compareTo("variableSymbol")
        == 0) {
        aVariableDeclarationList.add(
            new EstrelVariable(
                theparser,
                theSymbolTable
        );
    }
}
}
*/
* @since 1.7
 */
public final void parseEstrelStatement(final EstrelParser theparser)
throws EstrelParserException {
    ArrayList<VariableDeclaration> = new ArrayList();
    String theSymbolTable = "";
    List anElementList =

```

```

+ "[local-name(.)='VariableSymbol'][@id=" + aVariableElement.getAttributeValue("id") + "]/*");
}
else {
    throw new EsterelParserException(
        this.getEsterelStatementList.trimToSize() +
        + " statements incorrect",
        theparser);
}
//for
aVariableDeclarationList =
(EsterelVariable[]) aVariableDeclarationList
.toarray(new EsterelVariable[aVariableDeclarationList.size()]);
270
//if size=2
//public void parseEsterelStatement(final EsterelParser theparser)
//*
* Fills the substatements in the esterel module.
*<br> Sideffects: none
* Optram theparser the actual parser
* Optram EsterelParserException is only delivered
* @see kiel.fileInterface.estrel.EsterelParserException
* @see kiel.fileInterface.estrel1.EsterelParser
*
public final void setSubStatements(final EsterelParser theparser)
throws EsterelParserException {
int i = 0; // the index of the substatement in an ast file
List astElementList =
DONHelpers.getElements(
    theparser.getMLDocument(),
    this.getXPathSearchString());
this.setVariableStatement(theparser, getAddAt());
theparser.getEsterelModule(e(theparser.getactualEsterelModule()))
.addStatement(astModule(
    this.aStatement,
    theparser.createStatement((Element) (astElementList.get(i)))));
}
//Sets the index number from <code>EsterelModule</code>
2250
//public final String toString(final EsterelModule anEsterelModule)
{
    this.aStatement =
        theparser.createStatement(this.getEsterelStatementName());
}

```

C.2.32. LoopEachEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.32 dargestellt.

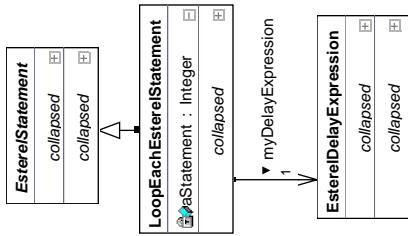


Abbildung C.25.: Klassendiagramm LoopEachEsterelStatement

```

Aufführung C.37: Die Klasse LoopEachEsterelStatement 20
package kiel.fileInterface.estrel.estrel2studio;
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.estrel.EsterelParser;
import kiel.util.DOMHelpers;
import kiel.util.StateChartHelpers;
/
import org.json.Element;
*/
* <p>Copyright: Copyright (c) 2004</p>
* <p>Company: Uni Kiel</p>
*
* @author <a href="mailto:l.kuehl@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 substratement in an
     * <code>EsterelModule</code>
     * @see kiel.fileInterface.estrel.EsterelModule
     */
    private int aStatement;
    /**
     * The value of the Expressions */
    private EsterelDelayExpression myDelayExpression;
    /**
     * Simple constructor.
     */
    public LoopEachEsterelStatement() {
        super();
    }
}

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

40     this.aStatement = -1;
        this.myDelayExpression = null;
    }
    /**
     * Parses a Document and set the class variables.
     * <br>It calls also the super constructor<code>EsterelStatement</code> methode
     * (EsterelParser theparser)</code> .
     * <br>Sideeffects:
     * Changes <code>EsterelParser</code>class variables
     * @param theparser an intace of a EsterelParser
     * @throws EsterelParserException is only delivered
     * @see kiel.fileInterface.estrel.EsterelParser
     * @see EsterelStatement#EsterelStatement(EsterelParser)
     * @see kiel.fileInterface.estrel.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 estrel statement.
     * @param anEsterelModule the estrel module which
     * becomes a KIEL Statchart
     * @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 theTreeTrapsSignals
     * Stack with all tree trap signals.
     * @throws Esterel2studioException
     * If the conversion to Kiel is not working. <br>
     * Sideffects : none
     * @return a State
     * @see kiel.fileInterface.estrel.estrel2estudio.EsterelModule
     * @see kiel.dataStructure.State convertTokiel(
     * final EsterelModule anEsterelModule,
     * final boolean isFinalState,
     * final ArrayList theLocalEvents,
     * final ArrayList theTreeTrapsSignals)
     * throws Esterel2studioException {
     * InitialArc alInitialArc = new InitialArc();
     * InitialState anInitialState = new InitialState();
     * State theStatementState =
     */
    public final kiel.dataStructure.State convertTokiel(
        final EsterelModule anEsterelModule,
        final boolean isFinalState,
        final ArrayList theLocalEvents,
        final ArrayList theTreeTrapsSignals)
        throws Esterel2studioException {
        InitialArc alInitialArc = new InitialArc();
        InitialState anInitialState = new InitialState();
        State theStatementState =
    }

    /**
     * (EsterelStatement)
     *
     * anEsterelModule.getEsterelProgram().get(
     *     this.aStatement)).convertTokiel(
     *     anEsterelModule,
     *     false,
     *     theLocalEvents,
     *     theLocalVariables,
     *     theTreeTrapsSignals);
     * // false means, substate is not a finalstate because its a loop
     * Transition aTransition =
     * StateChartHelpers.createSA(theStatementState, 1, theStatementState);
     * // the returned transition is never used,
     * aTransition.setLabel(
     *     this.myDelayExpression.convertTokiel(
     *         anEsterelModule,
     *         theLocalEvents,
     *         theLocalVariables,
     *         null));
     * alInitialArc.setSource(anInitialState);
     * alInitialArc.setTarget(theStatementState);
     * return StateChartHelpers.createOR(
     *     this.getEsterelStatementName() + "state",
     *     new Node[] {anInitialState, theStatementState});
    }

    /**
     * Implements the abstract methode from <code>EsterelStatement</code>
     * Parses an estrel statement in a <code>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.estrel.EsterelParserException
     * @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see EsterelStatement#parseEsterelStatement(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) {
    }
}

```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

C.2.33. LoopEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.33 dargestellt.

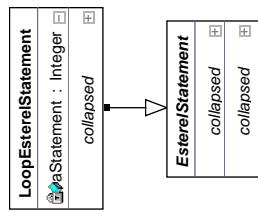


Abbildung C.26.: Klassendiagramm LoopEsterelStatement

Auflistung C.38: Die Klasse LoopEsterelStatement

```

package kiel.fileInterface.esterel.estrel2estudio;
// Revision 1.52 2005/11/10 12:26:26 1ku
// *** empty log message ***
30   *
import java.util.ArrayList;
* Revision 1.44 2005/10/04 11:33:29 1ku
import java.util.List;
* *** empty log message ***
* Revision 1.24 2005/05/12 01:37:57 1ku
import kiel.dataStructure.InitialArc;
* *** empty log message ***
import kiel.dataStructure.InitialState;
* Revision 1.9 2005/01/14 16:42:26 1ku
import kiel.dataStructure.Node;
* *** empty log message ***
import kiel.dataStructure.State;
* Revision 1.8 2005/01/12 15:00:00 1ku
import kiel.fileInterface.State;
* <br>
import kiel.fileInterface.esterel.EsterelParser;
* add Javadoc + check for project manual code conventions
import kiel.fileInterface.esterel.EsterelParserException;
* <br>
import kiel.util.DOHHelpers;
* fix Javadoc
import kiel.util.StateCharHelps;
* <br>
import org.json.Element;
* add Javadoc
import org.json.JSONObject;
* <br>
* <p> This is subclass of <code>EsterelStatement</code>
* classes implements the loop statement.</p>
* <p>Copyright: Copyright (c) 2004</p>
* <p>Company: Uni Kiel</p>
* <br>
* @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 1ku
* *** empty log message ***
50   *
public class LoopEsterelStatement extends EsterelStatement {
    /**
     * The index number of the substatement in an
     * <code>EsterelModule</code>
     * @see kiel.fileInterface.estrel.estrelModule
     */
    private int astatement;
}
  
```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

    /**
     * Simple constructor.
     */
    public LoopEsterelStatement() {
        super();
        //public LoopEsterelStatement() {
            /**
             * Parses a Document and set the class variables.
             * <br>It calls also the super constructor<code>EsterelStatement</code>
             * (EsterelParser theparser)</code> .
             * <br>Calls the <code>parseEsterelStatement</code> methode
             * Changes<code>EsterelParser</code>>class variables
             * @param theparser an instance of EsterelParser
             * throws EsterelParserException is only delivered
             * @see kiel.fileInterface.estrel.EsterelParserException
             * @see #parseEsterelStatement(EsterelParser)
             * @see EsterelStatement#parseEsterelStatement(EsterelParser)
             * @see kiel.fileInterface.estrel.EsterelParser
             */
            public LoopEsterelStatement(final EsterelParser theparser)
                throws EsterelParserException {
                    super(theparser);
                    this.parseEsterelStatement(theparser);
                }
            /**
             * Implements the methode from <code>EsterelStatement</code> .
             * Returns a state representation of the estrel statement
             * Returns an EsterelModule the estrel module which
             * becomes a KIEL Statechart
             * @param istinitialstate 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 theTreeTrapSignals
             * Stack with all tree trap signals.
             * @throws EsterelStateException
             * if the conversion to Kiel is not working. <br>
             * Sideeffects: none
             */
            public void parseEsterelStatement(final EsterelParser theparser)
                throws EsterelParserException {
                    List<Element> anElementList = null;
                    anElementList =
                        DMIElpers.getElements(
                            theparser.getXMLDocument(),
                            this.getXPathSearchString());
                    if (anElementList.size() == 1) {
                        theparser.incStatementCounter();
                    } else {
                        throw new EsterelParserException(
                            "this.getEsterelStatementName() + \" statement wrong \" , "
                            + theparser);
                    }
                }
            /**
             * Sets the index number from <code>EsterelModule</code>
             */
        }
    }
}

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

    theparser.getEsterelModule(
        theparser.getActualEsterelModule()
        .addStatementToModule(
            theparser.getAddit(),
            theparser.createStatement(((Element) (anElementList.get(i)))));
    }
}

/*
 * Implements the abstract methode from <code>EsterelStatement</code>.
 * Returns a string representation of an esterel statement.
 */
200 public final String toString() {
    /**
     * @param anEsterelModule an esterel modul representation
     * @return a String representation of an esterel statement
     * @see kiel.fileInterface.estrel.estrel2estudio.EsterelStatement
     * @see kiel.fileInterface.estrel.estrel2estudio.EsterelModule
     */
    public final String toString(final EsterelModule anEsterelModule) {
        return this.getEsterelStatementName()
            + "\n"
            + (
                (EsterelStatement) anEsterelModule.getEsterelProgram().get(
                    this.aStatement()).toString(
                        this.aStatement())
                + "end "
                + anEsterelModule)
            + this.getEsterelStatementName()
            + "\n";
    }
}

/*
 * //public final String toString(final EsterelModule anEsterelModule) {
 */
210 public final void setLoopStatement(final int aStatementIndex) {
    /**
     * Fills the statements in the esterel module.
     * @param aStatementIndex: changes EsterelModule
     */
    this.aStatement = aStatementIndex;
    setLoopStatement(final int aStatement);
}

/*
 * Fills the statements in the esterel module.
 * @param theparser: changes EsterelModule
 * @throws EsterelParserException is only delivered
 * @see kiel.fileInterface.estrel.EsterelParserException
 */
180 public final void setSubStatements(final EsterelParser theparser)
    throws EsterelParserException {
    int i = 0;
    List anElementList = null;
    anElementList =
        DOMEHelpers.getElements(
            theparser.getWIDocument(),
            this.getAPatchSearchString());
    if (anElementList.size() == 1) {
        this.setLoopStatement(theparser.getAddit());
    }
}

/*
 * of the substatement.
 * <br>Sideeffects:
 * Changes class variable <code>aStatement</code>
 * @param aStatementIndex an index of the statement in
 * an EsterelModule
 * @see kiel.fileInterface.estrel.estrel2estudio.EsterelModule
 */
170 public final void setLoopStatement(final int aStatementIndex) {
    /**
     * Fills the statements in the esterel module.
     * @param aStatementIndex: changes EsterelModule
     */
    this.aStatement = aStatementIndex;
    setLoopStatement(final int aStatement);
}

```

C.2.34. NothingEstrelStatement

Der Klassenaufbau ist in der Abbildung C.2.34 dargestellt.

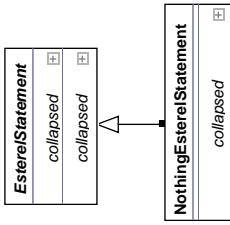


Abbildung C.27.: Klassendiagramm NothingEstrelStatement

Aufstellung C.39: Die Klasse NothingEstrelStatement

```

package kiel.fileInterface.estrel.estrel2estudio;
import java.util.ArrayList;
/
import kiel.dataStructure.FinalSimpleState;
import kiel.dataStructure.InitialArc;
import kiel.dataStructure.InitialState;
import kiel.dataStructure.Node;
import kiel.fileInterface.estrel.EstrelParser;
import kiel.fileInterface.estrel.EstrelParserException;
import kiel.util.StateChartHelpers;
/
/*
 * <p> This is a subclass of <code>EstrelStatement</code>
 * classes implements the nothing statement.</p>
 * <p>Copyright: Copyright (c) 2004</p>
 * <p>Company: Uni Kiel</p>
 */
20  * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kuehl</a>
* @version $Revision: 1.53 $
* $LastChangedDate: 2006/02/06 18:35:29 $
* $Log: NothingEstrelStatement.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
* Revision 1.43 2005/10/04 11:33:29 lku
30  * *** 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 ***
*
* Revision 1.7 2005/01/12 15:00:00 lku
* <br>
* add Javadoc + check for project manual code conventions
40  * <br>
* Revision 1.8 2005/01/14 15:00:00 lku
* <br>
* fix Javadoc
*/
50  public class NothingEstrelStatement extends EstrelStatement {
    /*
     * Simple constructor.
     */
    public NothingEstrelStatement() {
        super();
    }
    /**
     * Parses a Document and set the class variables.
     */
    /*
     * <br>It calls also the super constructor<code>EstrelStatement
     * (EstrelParser theparser)</code>.
     * <br>Calls the <code>parseEstrelStatement</code> methode
     */
}
  
```

C. Java Code für die Transformation von Esterel in SyncCharts

```

new Node[] {anInitialState, theEndState};

    } //public final kiel.dataStructure.State convertToKiel(
    /**
     * Implements the abstract methode from <code>EsterelStatement</code>
     * Parses an esterel statement in an <code>.jdom.Document</code>
     * representation of an .exp file
     */
    /* if you have not called the constructor
     * 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 a EsterelParser
    * @see #parseEsterelStatement(EsterelParser)
    * @see EsterelStatement#EsterelStatement(EsterelParser)
    * @see kiel.fileInterface.esterel.EsterelParser
    */
    public NothingEsterelStatement final EsterelParser theparser) {
        super(theparser);
        this.parseEsterelStatement(theparser);
        /*@see 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 theTreeTrapSignals .
        * Stack with all tree trap signals.
        */
        /*@throws Esterel2StudioException
        */
        * If the conversion to Kiel is not working. <br>
        * Sideeffects: none
        * @return a State
        * @see kiel.fileInterface.esterel.esterel2studio.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 theTreeTrapSignals)
            throws Esterel2StudioException {
            InitialArc aInitialArc = new InitialArc();
            InitialState anInitialState = new InitialState();
            FinalSimpleState theEndState = new FinalSimpleState();
            aInitialArc.setSource(anInitialState);
            aInitialArc.setTarget(theEndState);
            if (isFinalState) {
                return StateChartHelpers.createFinalORC(
                    this.getEsterelStatementName() + "state",
                    new Node[] {anInitialState, theEndState});
            }
            return StateChartHelpers.createOR(
                this.getEsterelStatementName() + "state",
                */
                new Node[] {anInitialState, theEndState});
        }
    }

    /**
     * Parses an esterel statement in an <code>.jdom.Document</code>
     * representation of an .exp file
     */
    /*@see EsterelStatement#parseEsterelStatement(EsterelParser)
     */
    public final void parseEsterelStatement(final EsterelParser theparser) {
        /*
         * Fills the substatements in the esterel module
         */
        * in this class it is empty.
        */
        /*@throws EsterelParserException
        */
        * @param theparser the actual parser
        * @throws EsterelParserException is only delivered
        * @see kiel.fileInterface.esterel.EsterelParserException
        * @since 1.9
        */
        public final void setSubStatements(final EsterelParser theparser)
            throws EsterelParserException {
        }

        /**
         * Implements the abstract methode from <code>EsterelStatement</code>.
         */
        /*@return a string representation of an esterel statement
         * @param anEsterelModule an esterel module representation
         * @return a String representation of an esterel statement
         * @see kiel.fileInterface.esterel.esterel2studio.EsterelStatement
         * @since 1.7
         */
        public final String toString(final EsterelModule anEsterelModule) {
            return this.getEsterelStatementName() + "\n";
        }
    }
}

```

C.2.35. ParallelEstrelStatement

Der Klassenaufbau ist in der Abbildung C.2.35 dargestellt.

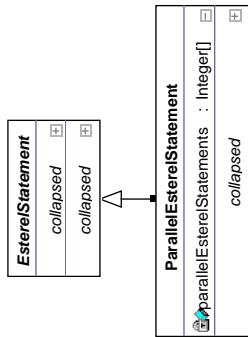


Abbildung C.28.: Klassendiagramm ParallelEstrelStatement

Auflistung C.40: Die Klasse ParallelEstrelStatement

```

package kiel.fileInterface.estrel.estrel2estudio;
/*
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.estrel.EstrelParser;
import kiel.fileInterface.estrel.EstrelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StatechartHelpers;
/*
import org.json.Element;
*/
/*
 * <p> This is a subclass of <code>EstrelStatement</code>
 * classes implements the parallel statement.</p>
 */
20   * <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: ParallelEstrelStatement.java,v $
   * Revision 1.55 2006/02/06 18:35:29 lku
   * private int[] parallelEstrelStatements;
   */

   * *** empty log message ***
   *
30   * 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 ***
   *
   * Revision 1.8 2005/01/14 15:00:00 lku
   * <br>
   * add Javadoc + check for project manual code conventions
   *
   * <br>
   * Revision 1.8 2005/01/14 15:00:00 lku
   * fix Javadoc
   */
50   public class ParallelEstrelStatement extends EstrelStatement {
   * ** The indexes numbers of the substatements in an
   * <code>EstrelModule</code>
   * @see kiel.fileInterface.estrel.estrel2estudio.EstrelModule
   */
  
```

C. Java Code für die Transformation von Esterel in SyncCharts

```

  /**
   * Simple constructor.
   */
  public ParallelEsterelStatement() {
    super();
  }

  /**
   * Parses a Document and set the class variables.
   * <br>It calls also the super constructor</code>EsterelStatement</code>
   * (EsterelParser theparser)</code> .
   * <br>Calls the <code>parseEsterelStatement</code> methode
   * <br>Sideeffects:
   * Changes <code>EsterelParser</code>class variables
   * @param theparser an instance of a EsterelParser
   * @see #parseEsterelStatement(EsterelParser)
   * @see EsterelStatement#EsterelStatement(EsterelParser)
   * @see kiel.file.Interface.estrel.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 estrel statement
   * @param anEsterelModule the estrel 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 theTreeTrapSignals
   * Stack with all tree trap signals.
   * @throws Esterel2StudioException
   * if the conversion to Kiel is not working. <br>
   * Sideeffects: none
   * @return a State
   * @see kiel.file.Interface.estrel.estrel2studio.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<theTreeTrapSignals>
  ) throws Esterel2StudioException {
    Region[] theRegions = new Region[this.parallelEsterelStatements.length];
    for (int i = 0; i < this.parallelEsterelStatements.length; i++) {
      InitialState initialState = new InitialState();
      InitialArc initialArc = new InitialArc();
      State aState =
        (
          (EsterelStatement)
        )
      anEsterelModule.getEsterelProgram().get(
        this.parallelEsterelStatements[i]);
      .convertToKiel(
        anEsterelModule,
        true,
        theLocalEvents,
        theLocalVariables,
        theTreeTrapSignals);
      // true means that the substate is final
      initialArc.setSource(anInitialState);
      initialArc.setTarget(aState);
      theRegions[i] =
        StatChartHelpers.createRegion(
          new Node[] {anInitialState, aState});
      } //for
      if (isFinalState) {
        return StatChartHelpers.createFinalAND(
          this.getEsterelStatementName() + "state",
          theRegions);
      }
      return StateChartHelpers.createAND(
        this.getEsterelStatementName() + "state",
        theRegions);
    }
  }
}

 /**
  * /public final kiel.dataStructure.State convertToKiel(
  /**
   * Implements the abstract methode from <code>EsterelStatement</code>.
   * Parses an estrel statement in an <code>org.json.Document</code>
   * representation of an exp file.
   * if you have not called the constructor
   * <code> EsterelStatement(EsterelParser theparser)</code>
   * use the methode <code>parseEsterelStatement</code>
   * before <code>parseEsterelStatement</code>
   * <br>Sideeffects:
   * changes <code>EsterelParser</code>class variables
   * @param theparser an instance of <code>EsterelParser</code>
   * @see EsterelStatement#EsterelStatement(EsterelParser)
   * @see org.json.Document
   */
  public final void parseEsterelStatement(final EsterelParser theparser) {
    List<anElementList =
      DOMHelpers.getElements(
        theparser.getXMLDocument(),
        this.getXPathSearchString());
    //stelle fast wieviele Statements noch existieren
    this.parallelEsterelStatements = new int[anElementList.size()];
    // und erhöhe die Anzahl der noch zu Parsenden Statements
    theparser.inStatementCounter(anElementList.size());
    /**
     * Sets the index number from <code>aStatement</code>
     * of the substatement.
     * <br>Sideeffects:
     * Changes class variable <code>aStatement</code>
  }
}

```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

C.2.36. PauseEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.36 dargestellt.

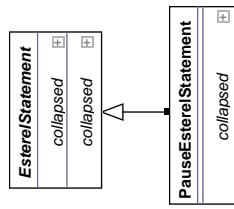


Abbildung C.29.: Klassendiagramm PauseEsterelStatement

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.SimpleTransition;
import kiel.dataStructure.eventesp.Tick;
import kiel.fileInterface.esterel.EsterelParser;
import kiel.fileInterface.esterel.EsterelParserException;
import kiel.util.StatechartHelpers;
import kiel.util.StatechartHelpers.*;
/**
 * <p> This is 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 1.53 2005/11/10 12:26:26 lku
 * *** empty log message ***
 * (EsterelParser theparser)</code>.
 */
public class PauseEsterelStatement extends EsterelStatement {
    /**
     * 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 16:42:26 lku
     * <br>
     * fix javadoc
     */
    public PauseEsterelStatement() {
        super();
    }
    /**
     * Simple constructor.
     */
    /**
     * Parses a Document and set the class variables.
     * <br>It calls also the super constructor<code>>esterelStatement
     */
}
  
```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

* <br>Calls the <code>parseEstrelStatement</code> methode
* <br>Sideeffects:
* Changes <code>EstrelParser</code> class variables
* @param theparser an instance of a EstrelParser
* @see #parseEstrelStatement(EstrelParser)
* @see EstrelStatement#EstrelStatement(EstrelParser)
* @see kiel.fileInterface.estrel.EstrelParser
*/
public PauseEstrelStatement(final EstrelParser theparser) {
    super(theparser);
    this.parseEstrelStatement(theparser);
}

/***
* Implements the methode from <code>EstrelStatement</code>.
* Returns a state representation of the estrel statement
* @param anEstrelModule the estrel 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 theTreeTrapSignals Stack with all tree trap signals.
* @throws Estrel2estudioException
* If the conversion to Kiel is not working. <br>
* @return a State
* @see kiel.fileInterface.estrel.estrel2estudio.EstrelModule
* @see kiel.dataStructure.State
*/
public final kiel.dataStructure.State convertToKiel(
    final EstrelModule anEstrelModule,
    final boolean isFinalState,
    final ArrayList<theLocalEvents>,
    final ArrayList<theLocalVariables>,
    final ArrayList<theTreeTrapSignals>
) throws Estrel2estudioException {
    int priority = 1;
    InitialArc initialArc = 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());
    initialArc.setSource(anInitialState);
    initialArc.setTarget(theStartState);
    if (isFinalState) {
        return StatechartHelpers.createFinalOR(
}
    }

    this.getEstrelStatementName() + "state",
    new Node[] {anInitialState, theStartState, theEndState});
    return StatechartHelpers.createR(
        this.getEstrelStatementName() + "state",
        new Node[] {anInitialState, theStartState, theEndState});
}

/**/
//public final kiel.dataStructure.State convertToKiel(
//    * Implements the abstract methode from <code>EstrelStatement</code>
//    * Parses an estrel statement in an <code>org.json.Document</code>
//    * representation of an .exp file
//    * if you have not called the constructor
//    * @code> EstrelStatement(EstrelParser theparser)</code>
//    * use the methode <code>preParseEstrelStatement</code>
//    * before <code>parseEstrelStatement</code>
//    * in this special case nothing has to be done.
//    * <br>Sideeffects:
//    * none
//    * Changes <code>EstrelParser</code> class variables
//    * @param theparser an instance of <code>EstrelParser</code>
//    * @see EstrelStatement#EstrelStatement(EstrelParser)
//    * @see kiel.fileInterface.estrel.EstrelParser
//    */
public final void parseEstrelStatement(final EstrelParser theparser) {
    /**
     * Fills the substatements in the estrel module
     * in this class it is empty.
     * <br> Sideeffects: none
     */
    * Implements the abstract methode from <code>EstrelStatement</code>.
    * @param theparser the actual parser
    * @throws EstrelParserException is only delivered
    * @see kiel.fileInterface.estrel.EstrelParserException
    */
    public final void setSubStatements(final EstrelParser theparser)
throws EstrelParserException {
    /**
     * Implements the abstract methode from <code>EstrelStatement</code>.
     * Returns a string representation of an estrel statement.
     * @param anEstrelModule an estrel module representation
     * @return a String representation of an estrel statement
     * @see kiel.fileInterface.estrel.EstrelStatement
     * @see kiel.fileInterface.estrel2estudio.EstrelModule
     */
    public final String toString(final EstrelModule anEstrelModule) {
        return this.getEstrelStatementName();
    }
    //public final String toString(final EstrelModule aEstrelModule) {
}

}

```

C.2.37. PresentThenElse

Auflistung C.42: Die Klasse PresentThenElse

```

package kiel.fileinterface.estrel.estrel2studio;
/*
 * <br>
 * change Choice to DynamicChoice 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 statement in an
     * <code>estrelModule</code>
     * @see kiel.fileInterface.estrel.estrel2studio.EsterelModule
     */
    private int elsePartStatement;
    /**
     * the index numbers of the then part statements in an
     * <code>estrelModule</code>
     * @see kiel.fileInterface.estrel.estrel2studio.EsterelModule
     */
    private int[] theStatements;
    /**
     * Simple constructor.
     */
    public PresentThenElse() {
        super(null);
    }
    /**
     * Parses a Document and set the class variables.
     * <br>It calls also the super constructor<code>EsterelStatement
     * (EstrelParser theparser)</code>;
     * <br>Calls the <code>parseEsterelStatement</code> methode
     * Changes <code>EstrelParser</code> class variables
     * @param theparser an instance of a EstrelParser
     * @throws EstrelParserException is only delivered
     * @see kiel.fileInterface.estrel.EsterelParserException
     * @see #parseEsterelStatement(EstrelParser)
     * @see EsterelStatement(EsterelParser)
     * @see kiel.fileInterface.estrel.EsterelParser
     */
    public PresentThenElse(final EstrelParser theparser)
        throws EstrelParserException {
        super(theparser);
        this.theStatements = null;
        this.elsePartStatement = -1;
        this.aSignalExpression = null;
        if (theparser != null) {
            this.parseEsterelStatement(theparser);
        }
    }
    /**
     * Implements the methode from <code>EsterelStatement</code>.
     * Returns a state representation of the estrel statement
     * fix javadoc
     */
    //public PresentThenElse(final EsterelStatement theparser) {
    /**
     * Implements the methode from <code>EsterelStatement</code>.
     * Returns a state representation of the estrel statement
     * fix javadoc
     */
}

/*
 * This is a subclass of <code>EsterelStatement</code>
 * classes. It parses the estrel 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>
 * Revision $Revision: 1.56 $ last modified $Date: 2006/06/18:35:30 $
 * <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 1.8 2005/01/12 15:00:00 lku
 * <br>
 * Revision 1.9 2005/01/14 16:42:26 lku
 * *** empty log message ***
 * Revision 1.9 2005/01/14 16:00:00 lku
 * <br>
 * add javadoc + check for project manual code conventions
 * Revision 1.9 2005/01/14 16:00:00 lku
 * <br>
 * fix javadoc
 */

```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

    * @param anEstrelModule the estrel 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 theTreeTrapSignals
    * Stack with all tree trap signals.
    * @throws Estrel2EstudioException
    * if the conversion to Kiel is not working. <br>
    * Sideeffects: none
    * @return a State
    * @see kiel.fileInterface.estrel.estrel2estudio.EstrelModule
    * @see kiel.dataStructure.State
    */
    public final kiel.dataStructure.State convertToKiel(
        final EstrelModule anEstrelModule,
        final boolean isFinalState,
        final ArrayList<theLocalEvents>,
        final ArrayList<theLocalVariables>,
        final ArrayList<theTreeTrapSignals>)
        throws Estrel2EstudioException {
        InitialArc aInitialArc = new InitialArc();
        InitialState anInitialState = new InitialState();
        DynamicChoice aPresentTest = new DynamicChoice();
        int n0EndStates = this.aSignalExpression.length + 1;
        State[] theEndStates = new State[n0EndStates];
        // 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] =
                    (EstrelStatement)
                        anEstrelModule.getEstrelProgram().get(
                            .theStatements[theCaseIndex]));
            }
        }
        return anEstrelModule.convertToKiel(
            true,
            theLocalEvents,
            theLocalVariables,
            theTreeTrapSignals);
    }

    /**
     * Implements the abstract methode from <code>EstrelStatement</code>.
     * Parses an estrel statement in an <code>.org.jdom.Document</code>
     * representation of an .exp file.
     * If you have not called the constructor
     * <code>EstrelStatement(EstrelParser theparser)</code>
     * use the methode <code>preParseEstrelStatement</code>.
     * before <code>parseEstrelStatement</code>.
     * <br>Sideeffects:
     * the class variables are set
    */
    protected void convertToKiel(
        theLocalEvents,
        theLocalVariables,
        theTreeTrapSignals,
        ConditionalTransition aChoiceTransition =
            new ConditionalTransition();
        StateChartHelpers.set(
            aChoiceTransition,
            aPresentTest,
            n0EndStates,
            theEndStates[n0EndStates - 1]);
        theNodes[theNodes.length - 2] = anInitialState;
        theNodes[theNodes.length - 1] = aPresentTest;
        if (isFinalState) {
            return StateChartHelpers.createFinalOR(
                this.getEstrelStatementName() + "state",
                theNodes);
        }
        return StateChartHelpers.createOR(
            this.getEstrelStatementName() + "state",
            theNodes);
    }

    /**
     * Parses an estrel statement in an <code>.org.jdom.Document</code>
     * representation of an .exp file.
     * If you have not called the constructor
     * <code>EstrelStatement(EstrelParser theparser)</code>
     * use the methode <code>preParseEstrelStatement</code>.
     * before <code>parseEstrelStatement</code>.
     * <br>Sideeffects:
     * the class variables are set
    */
    protected void convertToKiel(
        theLocalEvents,
        theLocalVariables,
        theTreeTrapSignals,
        ConditionalTransition aChoiceTransition =
            new ConditionalTransition();
        StateChartHelpers.set(
            aChoiceTransition,
            aPresentTest,
            theCaseIndex + 1,
            theEndStates[theCaseIndex]);
        aChoiceTransition.setLabel(
            theCaseIndex);
    }
}

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

* Changes <code>EsterelParser</code> class variables
* @param theparser an instance of <code>EsterelParser</code>
* @throws EsterelParserException is only delivered
* @see kiel.fileInterface.estrel.EsterelParserException
220 * @see EsterelStatement#EsterelStatement(EsterelParser)
* @see kiel.fileInterface.estrel.EsterelParser
* @see org.json.Document
*/
public final void parseEsterelStatement(final EsterelParser theparser)
throws EsterelParserException {
    List<XMLElement> anElementList =
        DOMHelpers.getElements(
            theparser.getXMLDocument());
    anElementList.add(
        this.getXPathSearchString());
    int numberOfPredicated = 0;
    if (((Element) anElementList.get(anElementList.size() - 1)).getName()
        == "PredicatedStatement") {
        numberOfPredicated = anElementList.size();
    } else {
        numberOfPredicated = anElementList.size() - 1;
        theparser.increaseStatementCounter();
    }
    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.get XPathSearchString()
                    + "[@local-name(.) = '"
                    + ((Element) anElementList.get(theCaseIndex)).getName()
                    + "'] [!@id= ''"
                    + (
                        (Element) anElementList.get(
                            theCaseIndex)).getId());
            aPredicatedStatementXPathSearchString =
                aPredicatedStatementXPathSearchString
                    + "[!@id]";
            List<XMLElement> anElementList =
                DOMHelpers.getElements(
                    theparser.getXMLDocument(),
                    aCaseStatementCounter++);
            if (aPredicatedElementList.size() > 1) {
                aCaseStatementCounter++;
            }
            int index = aPredicatedElementList.size() - 1;
            // the SignalExpression is the second list element
            String aSubPredicatedStatementXPathSearchString =
280
                "/*[@local-name(.) = '" +
                    + ((Element) aPredicatedElementList.get(index)).getName() +
                    + "'] *";
            this.aSignalExpression[theCaseIndex] =
                new EsterelSignalExpression(
                    theparser,
                    aSubPredicatedStatementXPathSearchString,
                    theparser.getXMLOutputDocument());
        } else {
            throw new EsterelParserException(
                this.getEsterelStatementName() + " wrong statement",
                theparser);
        } //else if size*
    }
    theparser.increaseStatementCounter(aCaseStatementCounter);
}
//public final void parseEsterelStatement(final EsterelParser theparser)
//    //final EsterelModule anEsterelModule)
/*
 * Fills the substatements in the estrel module
 */
290
    /*<br> Sideffects: changes EsterelModule
     * @param theparser the actual parser
     * @throws EsterelParserException is only delivered
     * @see kiel.fileInterface.estrel.EsterelParserException
     * @see kiel.fileInterface.estrel.EsterelParser
     * @see kiel.fileInterface.estrel.estrel2estudio.EsterelModule
     */
    public final void setSubStatements(final EsterelParser theparser)
throws EsterelParserException {
    List<XMLElement> anElementList =
        DOMHelpers.getElements(
            theparser.getXMLDocument()),
    this.get XPathSearchString());
    for (int theCaseIndex < anElementList.size();
        theCaseIndex++) {
        if (((Element) anElementList.get(theCaseIndex)).getName()
            == "PredicatedStatement") {
            String aPredicatedStatementXPathSearchString =
                this.get XPathSearchString()
                    + "[@local-name(.) = ''" +
                    + ((Element) anElementList.get(theCaseIndex)).getName() +
                    + "[!@id= ''"
                    + (
                        (Element) anElementList.get(
                            theCaseIndex)).getId());
            theparser.setElements(
                theparser.getXMLDocument(),
                aCaseStatementCounter++);
        }
    }
}
300
310
320
240
250
260
270

```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

+ "      case "
+ "        + this.asSignalExpression(theCaseIndex]
+ "      + this.asSignalExpression(theCaseIndex]
+ "      .toString(anEsterelModule);
if (this.theStatements[theCaseIndex] == -1) {
  if (this.theStatements[theCaseIndex] != -1) {
    result += " do "
    + (
      (EsterelStatement) anEsterelModule
      .getEsterelProgram()
      .get(
        this.theStatements[theCaseIndex])).toString(
      anEsterelModule);
  }
}
else {
  if (Element) (aPredicatedElementList.get(0)));
  this.theStatements[theCaseIndex] = -1;
}
else {
  this.alsoParStatement = theparser.getAddAt();
  theparser.getEsterelModule()
  theparser.getActualEsterelModule().addStatementToModule(
  theparser.getAddAt(),
  theparser.createStatement(
  (Element) (anElementList
  .get(anElementList.size() - 1))));

}
// set
} // for
/** * Implements the abstract method from <code>EsterelStatement</code>. * Returns a string representation of an esterel statement. * <br>>Side effects: * none * param anEsterelModule an esterel module representation * return a string representation of an esterel statement * <code>kiel.fileInterface.esterel.esterel2estudio.EsterelStatement</code> * see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement */
public final String toString(final EsterelModule anEsterelModule) {
  String result = this.getEsterelStatementName();
  if (this.asSignalExpression.length > 1) {
    for (int theCaseIndex = 0;
    theCaseIndex < this.asSignalExpression.length;
    theCaseIndex++) {
      result += "\n" + this.getEsterelStatementName();
    }
  }
}

```

C.2.38. ProcedureCallEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.38 dargestellt.

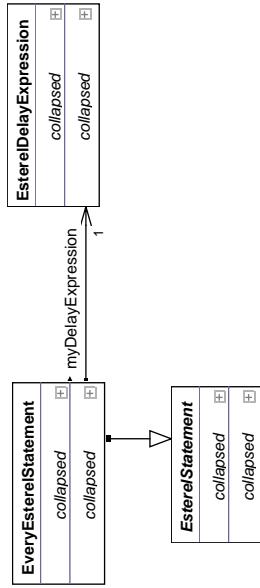


Abbildung C.30.: Klassendiagramm ProcedureCallEsterelStatement

Auflistung C.43: Die Klasse ProcedureCallEsterelStatement

```


    /**
     * the procedure name.
     */
    private EsterelProcedureDeclaration myProcedureDeclaration = null;
    /**
     * the reference parameters.
     */
    private EsterelVariable[] myReferenceParameter;
    /**
     * the value parameters.
     */
    private EsterelExpression[] myValueParameter;
    /**
     * simple constructor.
     */
    public ProcedureCallEsterelStatement() {
        super();
        this.myReferenceParameter = null;
        this.myValueParameter = null;
    }
}


    /**
     * 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 {


```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

* <br>Sideeffects:
* Changes <code>EstrelParser</code>-class variables
* @param theparser an intace of a EstrelParser
* @throws EstrelParserException
* Only delivered
* @see kiel.fileInterface.estrel.EstrelParser
* @see #parseEstrelStatement(EstrelParser)
* @see EstrelStatement#EstrelParser
* @see kiel.fileInterface.estrel.EstrelParser
*/
60 public ProcedureCallEstrelStatement(final EstrelParser theparser)
throws EstrelParserException {
    super(theparser);
    this.myReferenceParameter = null;
    this.myValueParameter = null;
    if (this.getPath searchString() != null && theparser != null) {
        this.parseEstrelStatement(theparser);
    } else {
        throw new EstrelParserException(
            "ProcedureCall : no parser or path");
    }
}
/**
* Implements the methode from <code>EstrelStatement</code>.
*
* Returns a state representation of the estrel statement
* @param anEstrelModule the estrel 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 theTrapSignals Stack with all tree trap signals.
* @throws Estrel2estudioException
* @param if the conversion to Kiel is not working. <br>
* Sideeffects: none
* @return a State
* @see kiel.fileInterface.estrel.estrel2estudio.EstrelModule
* @see kiel.dataStructure.State
*/
90 public final kiel.dataStructure.State convertToKiel(
    final EstrelModule anEstrelModule,
    final boolean isFinalState,
    final ArrayList<theLocalEvents>,
    final ArrayList<theLocalVariables>,
    final ArrayList<theTrapSignals>)
throws Estrel2estudioException {
    InitialArc initialArc = new InitialArc();
    InitialState anInitialState = new InitialState();
    FinalSimpleState theEndState = new FinalSimpleState();
    initialArc.setSource(anInitialState);
    initialArc.setTarget(theEndState);
    StringLabel aLabel = new StringLabel("// " + this.toString(anEstrelModule));
    if (isFinalState) {
        initialArc.setLabel(aLabel);
        return StatechartHelpers.createFinalOR(
    }
}

110 return StatechartHelpers.createOR(
    this.getEstrelStatementName() + "state",
    new Node[] {anInitialState, theEndState});
}
120 /**
* Implements the abstract methode from <code>EstrelStatement</code>
* Parses an estrel statement in an <code>org.jdom.Document</code>
* representation of an .exp file
* If you have not called the constructor
* use the methode <code>parseEstrelStatement</code>
* before <code>parseEstrelStatement</code>.
* <br>Sideeffects:
* the class variables are set
* Changes <code>EstrelParser</code>-class variables
* @param theparser an instance of <code>EstrelParser</code>
* @throws EstrelParserException is only delivered
* @see kiel.fileInterface.estrel.EstrelParserException
* @see EstrelStatement#parseEstrelStatement(EstrelParser)
* @see EstrelStatement#parseEstrelStatement(EstrelParser)
* @see kiel.fileInterface.estrel.EstrelParser
* @see org.jdom.Document
*/
130 public final void parseEstrelStatement(final EstrelParser theparser)
throws EstrelParserException {
    List<DMHelpers.getELement> =
        DMHelpers.getELementList(
            this.getXPathSearchString(),
            this.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
            theparser =
                .getEstrelModule(theparser.getActualEstrelModule());
            theparser.getProcedureID();
            anElement.getAttributeValue("id");
        } else if (theEoVCounter == 0) {
            theReferenceParameter.add(
    }
}
150 /**
* @param if (theEoVCounter == 0) {
            theReferenceParameter.add(
    }
}
160

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

    * Returns a string representation of an esterel statement.
    * @param anEsterelModule an esterel modul representation
    * @return a String representation of an esterel statement
    * @see kiel.fileInterface.esterel.esterel2estudio.EsterelStatement
    * @see kiel.fileInterface.esterel.esterel2estudio.EsterelModule
    */
    public final String toString(final EsterelModule anEsterelModule) {
        210       String result = "call \"";
        /* 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()
                    + Esterel2studioProperties.getSeparatorString();
            }
            result += this.myReferenceParameter[2]
                .getVariableIdentifier();
        }
        result += " ";
        if (this.myValueParameter != null
            && this.myValueParameter.length > 0) {
            for (int i = 0; i < this.myValueParameter.length - 1; i++) {
                result += this.myValueParameter[i];
            }
            result += " ";
        }
        result += this.myValueParameter[2]
            .toString(anEsterelModule);
        result += " ";
        return result;
    }

    /**
     * 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 parser the actual parser
     * @throws EsterelParserException is only delivered
     * @see kiel.fileInterface.esterel.EsterelParserException
     * @see kiel.fileInterface.esterel.EsterelParser
     */
    public final void setSubStatements(final EsterelParser theparser)
        throws EsterelParserException {
        200       /*Implements the abstract methode from <code>EsterelStatement</code>.
    */

```

C.2.39. RepeatEstrelStatement

Der Klassenaufbau ist in der Abbildung C.2.39 dargestellt.

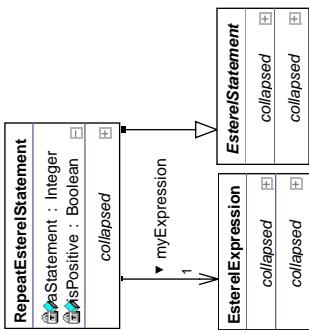


Abbildung C.31.: Klassediagramm RepeatEstrelStatement

Auflistung C.44: Die Klasse RepeatEstrelStatement

```

package kiel.fileInterface.estrel.estrel2estudio;

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.integx.IntegerVariable;
import kiel.fileInterface.estrel.EstrelParser;
import kiel.fileInterface.estrel.EstrelParserException;
import kiel.util.DOMHelpers;

import org.jdom.Element;

import kiel.util.StatechartHelpers;

public class RepeatEstrelStatement extends EstrelStatement {
    /**
     * This is a subclass of <code>EstrelStatement</code>
     * classes implements the await statement.
     */
    /**
     * It implements await s,await s do p; await case s do p; await case s
     * <p>
     * <p>
     * Copyright : Copyright (c) 2004
     * </p>
     * <p>
     * Company : Uni Kiel
     * </p>
     * <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 void accept(StatechartHelpers helper) {
        helper.visit(this);
    }
}
  
```

C. Java Code für die Transformation von Esterel in SyncCharts

```

* <code>EsterelModule</code>
50      *
      * @see kiel.fileInterface.estrel.estrel2studio.EsterelModule
      */
    private int aStatement;

    /**
     * true if the repeat is positive.
     */
    private boolean isPositive;

60    /**
     * 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 variables. <br>
     * It calls also the super constructor <code>EsterelStatement</code><br>
     * (EsterelParser theParser)</code>.
     */
    public void parseEsterelStatement(<code> methodo <br>
80      * Sideffects: Changes <code>EsterelParser</code> class variables
      * @param theparser
      * an interface of a EsterelParser
      * @throws EsterelParserException
      * @see kiel.fileInterface.estrel.EsterelParserException
      * @see #parseEsterelStatement(EsterelParser)
      * @see EsterelStatementEsterelStatement(EsterelParser)
      * @see kiel.fileInterface.estrel.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 estrel statement.
     */
100   /**
     * Returns a state representation of the estrel statement.
     */

```

* ©Param anEsterelModule
 * the estrel 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 theTreeTrapSignals
 * Stack with all tree trap signals.
 * @throws Esterel2StudioException
 * If the conversion to Kiel is not working.

 * Sideffects: none
 * @return a State
 * @see kiel.fileInterface.estrel.estrel2studio.EsterelModule
 * @see kiel.dataStructure.State
 */
 public final kiel.dataStructure.State convertToKiel(
 final EsterelModule anEsterelModule,
 final boolean isFinalState,
 final ArrayList<theLocalEvents> theLocalEvents,
 final ArrayList<theLocalVariables> theLocalVariables,
 final ArrayList<theTreeTrapSignals> theTreeTrapSignals)
 throws Esterel2StudioException {
 // create two variables
 EsterelExpression oneTimeEx = new EsterelExpression();
 oneTimeEx.settype(anEsterelModule.getEsterelTypeByName("integer"));
 oneTimeEx.setLiteral("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));
 }

 /**
 * Converts a KIEL Statechart to an Esterel module.
 */
150 /**
 * Converts a KIEL Statechart to an Esterel module.
 */

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

theLocalVariables.add((anEstereModule
    .getEstereVariableID(oneTime.getXMLID())));
    convertToKiel(anEstereModule,
        theLocalEvents,
        theLocalVariables));
    theLocalVariables.add((anEstereModule
        .getEstereVariableID(theTimes.getXMLID())));
    convertToKiel(anEstereModule,
        theLocalEvents,
        theLocalVariables));
    DynamicChoice dc = new DynamicChoice();
    FinalSimpleState theEndState = new FinalSimpleState();
    initialState.anInitialState = new InitialState();
    State theStatementState = ((State)statement)
        (anEstereModule.getEstereProgram()
            .get(this.s.aStatement())));
    convertToKiel(anEstereModule,
        false,
        theLocalEvents,
        theLocalVariables,
        theTrapSignals);
    // false means, substrate is not a finalstate because its a loop
    IntegerVariable v = (IntegerVariable) (anEstereModule
        .getEstereVariableID(theTimes.getXMLID()))
    .convertToKiel(anEstereModule,
        theLocalEvents,
        theLocalVariables);
    IntegerVariable e = (IntegerVariable) (anEstereModule
        .getEstereVariableID(oneTime.getXMLID()));
    convertToKiel(anEstereModule,
        theLocalEvents,
        theLocalVariables);
    int priority = 1;
    if (this.isPositive) {
        InitialArc timesRepeatArc = new InitialArc();
        timesRepeatArc.setSource(initialState);
        CompoundLabel theCLabel = new CompoundLabel();
        theCLabel.setCondition(StateChartHelpers
            .getLabel(anEstereModule.getEstereStatechart(),
                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(theEndState);
    defaultArc.setPriority(2);
    NormalTermination theNT = StateChartHelpers.createNT(theStatementState,
        1,
        210
    )
    TransitionLabel theLabel = null;
    Actions theActions =
        StateChartHelpers
            .getActions(anEstereModule
                .getEstereStatechart(),
                    theLocalVariables,
                    theLocalEvents,
                    e.getName()
                    + "!="
                    + e.getName())
                    if (theActions == null) {
                        theLabel = new StringLabel("//"
                            + e.getName()
                            + "!="
                            + e.getName()
                            + "!="
                            + e.getName());
                    } else {
                        theLabel = new CompoundLabel();
                        ((CompoundLabel) theLabel).setEffect(theActions);
                    }
                    theLabel.setLabel(theLabel);
                    ConditionalTransition endrepeat = new ConditionalTransition();
                    endrepeat.setDef(new ConditionalTransitionDef(
                        endrepeat.setSource(dc),
                        def.setSource(dc),
                        endrepeat.setTarget(theEndState),
                        def.setTarget(theStatementState),
                        endrepeat.setPriority(1),
                        CompoundLabel acL = new CompoundLabel();
                        acL.setCondition(StateChartHelpers
                            .getLabel(anEstereModule.getEstereStatechart(),
                                theLocalVariables,
                                theLocalEvents,
                                v.getName() + "<="));
                        if (acL.getCondition() == null) {
                            endrepeat.setLabel(new StringLabel(
                                v.getName() + "<="
                                + e.getName())));
                        } else {
                            endrepeat.setLabel(acL);
                        }
                    );
                    theLocalVariables.remove(theLocalVariables.size() -
                        theLocalVariables.size());
                    result.addNode(dc);
                    result.addNode(theStatementState);
                    result.addNode(anInitialState);
                    result.addNode(theEndState);
                    anInitialState.setParent(result);
                    theStatementState.setParent(result);
                    dc.setParent(result);
                    return result;
    220
    230
    240
    250
    260
    200
    190
    180
    170

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

    }
}

/***
 * Implements the abstract methode from <code>EsterelStatement</code>
 * Parses an esterel statement in an <code>org.json.Document</code>
 * representation of an .exp file if you have not called the
 * constructor <code> EsterelStatement(EsterelParser theparser)</code>
280   * use the methode <code>preParseEsterelStatement</code> before
 * <code>parseEsterelStatement</code> <br>
 * Sideseffects: 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 EsterelStatement#parseEsterelStatement(EsterelParser)
 * @see EsterelStatement#parseEsterelStatement(EsterelParser)
 * @see org.json.Document
 */
public final void parseEsterelStatement(
    final EsterelParser theparser)
    throws EsterelParserException {
    final int afOUR = 4;
    List anElementList = null;
    anElementList =
        D0NHelpers.getElements(theparser.getXMLDocument(),
            this.getPathsSearchString());
    if (anElementList.size() == afOUR) {
        this.astatement = theparser.getAddAt();
        theparser.getEsterelModule(theparser.getActualEsterelModule())
            .addStatementToModule(theparser.getAddAt(),
                theparser
                    .createStatement(((Element) anElementList.get(1))));
    }
}

/***
 * Implements the abstract methode from <code>EsterelStatement</code>.
 * Returns a string representation of an esterel statement.
 * @param anEsterelModule
 *          an esterel modul representation
 */
300   public final String toString(
    final EsterelModule anEsterelModule) {
    String result = "";
    if (this.isPositive) {
        result += "positive ";
    }
    result += this.getEsterelStatementName() + " "
        + this.myExpression.toString(anEsterelModule)
        + " times \\" + " "
        + ((EsterelStatement) anEsterelModule).getEsterelProgram()
            .get(this.astatement).toString(anEsterelModule)
        + "end "
}

/*
 * Implements the abstract methode from <code>EsterelStatement</code>.
 * Returns a string representation of an esterel statement.
 * @param anEsterelModule
 *          an esterel modul representation
 */
310   public final String toString(
    final EsterelModule anEsterelModule) {
    String result = "";
    if (this.isPositive) {
        result += "positive ";
    }
    result += this.getEsterelStatementName() + " "
        + this.myExpression.toString(anEsterelModule)
        + " times \\" + " "
        + ((EsterelStatement) anEsterelModule).getEsterelProgram()
            .get(this.astatement).toString(anEsterelModule)
        + "end "
}

/*
 * Implements the abstract methode from <code>EsterelStatement</code>.
 * Parses an esterel statement in an <code>org.json.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>
 * Sideseffects: the class variables are set Changes
 * <code>EsterelParser</code> class variables
 * @param theparser
 *          the actual parser
 * @throws EsterelParserException
 *          is only delivered
 * @see kiel.fileInterface.estrel.EsterelParserException
 */
320   public final void setSubStatements(
    final EsterelParser theparser)
    throws EsterelParserException {
    int i = 0;
    final int afOUR = 4;
    List anElementList =
        D0NHelpers.getElements(theparser.getXMLDocument(),
            this.getPathsSearchString());
    if (anElementList.size() == afOUR) {
        this.astatement = theparser.getAddAt();
        theparser.getEsterelModule(theparser.getActualEsterelModule())
            .addStatementToModule(theparser.getAddAt(),
                theparser
                    .createStatement(((Element) anElementList.get(i))));
    }
}

/*
 * Implements the abstract methode from <code>EsterelStatement</code>.
 * Returns a string representation of an esterel statement.
 * @param anEsterelModule
 *          an esterel modul representation
 */
330   public final String toString(
    final EsterelModule anEsterelModule) {
    String result = "";
    if (this.isPositive) {
        result += "positive ";
    }
    result += this.getEsterelStatementName() + " "
        + this.myExpression.toString(anEsterelModule)
        + " times \\" + " "
        + ((EsterelStatement) anEsterelModule).getEsterelProgram()
            .get(this.astatement).toString(anEsterelModule)
        + "end "
}

```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```
    + this.getEstrelStatementName() ;  
    return result + "\n";  
}
```

C.2.40. SequenceEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.40 dargestellt.

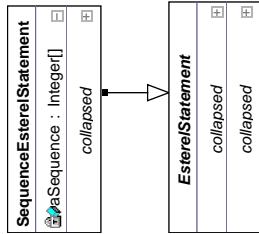


Abbildung C.32.: Klassendiagramm SequenceEsterelStatement

Auflistung C.45: Die Klasse SequenceEsterelStatement

```

package kiel.fileInterface.esterel.esterel2estudio;
/** The indexes numbers of the substatements in an
 * <code>esterelModule</code>
 * @see kiel.fileInterface.estrel.estrel2estudio.EsterelModule
30   */
private int[] aSequence;
/** 
 * Simple constructor.
 */
public SequenceEsterelStatement() {
    super();
}
/** 
 * Parses a Document and set the class variables.
 * <br>It calls also the super constructor<code>EsterelStatement</code>.
 * (EstrelParser theparser)</code>.
 * <br>Calls the <code>parseEsterelStatement</code> methode
 * <br>Sideeffects:
 * Changes <code>EsterelParser</code> class variables
 * @param theparser an intice of a EsterelParser
 * @see #parseEsterelStatement(EsterelParser)
 * @see EsterelStatement(EsterelParser)
 * @see kiel.fileInterface.estrel.estrel2estudio.EsterelModule
 */
public SequenceEsterelStatement#EsterelStatement(EsterelParser)
super(theparser);
this.parseEsterelStatement(theparser);
    */
public SequenceEsterelStatement#EsterelStatement(final EsterelParser theparser) {
    */
}
// public SequenceEsterelStatement extends EsterelStatement {
50
}

```

10 import kiel.util.DOMHelpers;

15 import kiel.util.StateCharHelps;

20 import org.json.Element;

25 /*
 * 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>
 * <p>Author Lars Kuehl
 * <p>Revision: 1.58 \$ last modified \$Date: 2006/02/06 18:35:30 \$
 *

 */
public class SequenceEsterelStatement extends EsterelStatement {

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

* 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 theTrapSignals Stack with all tree trap signals.
* @throws EsterelIOException
* if the conversion to Kiel is not working. <br>
* Sideeffects: none
* @return a State
* @see kiel.kielInterface.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<theTrapSignals>
) throws Esterel2EstdiException {
    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,
                            theTrapSignals);
        } //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,
                            theTrapSignals);
        } //true means, that this substate isfinal
    }
}
90
100
110
120
130
140
150
160
170
180
190
200
210
220
230
240
250
260
270
280
290
300
310
320
330
340
350
360
370
380
390
400
410
420
430
440
450
460
470
480
490
500
510
520
530
540
550
560
570
580
590
600
610
620
630
640
650
660
670
680
690
700
710
720
730
740
750
760
770
780
790
800
810
820
830
840
850
860
870
880
890
900
910
920
930
940
950
960
970
980
990
1000
1010
1020
1030
1040
1050
1060
1070
1080
1090
1100
1110
1120
1130
1140
1150
1160
1170
1180
1190
1200
1210
1220
1230
1240
1250
1260
1270
1280
1290
1300
1310
1320
1330
1340
1350
1360
1370
1380
1390
1400
1410
1420
1430
1440
1450
1460
1470
1480
1490
1500
1510
1520
1530
1540
1550
1560
1570
1580
1590
1600
1610
1620
1630
1640
1650
1660
1670
1680
1690
1700
1710
1720
1730
1740
1750
1760
1770
1780
1790
1800
1810
1820
1830
1840
1850
1860
1870
1880
1890
1900
1910
1920
1930
1940
1950
1960
1970
1980
1990
2000
2010
2020
2030
2040
2050
2060
2070
2080
2090
2100
2110
2120
2130
2140
2150
2160
2170
2180
2190
2200
2210
2220
2230
2240
2250
2260
2270
2280
2290
2300
2310
2320
2330
2340
2350
2360
2370
2380
2390
2400
2410
2420
2430
2440
2450
2460
2470
2480
2490
2500
2510
2520
2530
2540
2550
2560
2570
2580
2590
2600
2610
2620
2630
2640
2650
2660
2670
2680
2690
2700
2710
2720
2730
2740
2750
2760
2770
2780
2790
2800
2810
2820
2830
2840
2850
2860
2870
2880
2890
2900
2910
2920
2930
2940
2950
2960
2970
2980
2990
2995
3000
3005
3010
3015
3020
3025
3030
3035
3040
3045
3050
3055
3060
3065
3070
3075
3080
3085
3090
3095
3100
3105
3110
3115
3120
3125
3130
3135
3140
3145
3150
3155
3160
3165
3170
3175
3180
3185
3190
3195
3200
3205
3210
3215
3220
3225
3230
3235
3240
3245
3250
3255
3260
3265
3270
3275
3280
3285
3290
3295
3300
3305
3310
3315
3320
3325
3330
3335
3340
3345
3350
3355
3360
3365
3370
3375
3380
3385
3390
3395
3400
3405
3410
3415
3420
3425
3430
3435
3440
3445
3450
3455
3460
3465
3470
3475
3480
3485
3490
3495
3500
3505
3510
3515
3520
3525
3530
3535
3540
3545
3550
3555
3560
3565
3570
3575
3580
3585
3590
3595
3600
3605
3610
3615
3620
3625
3630
3635
3640
3645
3650
3655
3660
3665
3670
3675
3680
3685
3690
3695
3700
3705
3710
3715
3720
3725
3730
3735
3740
3745
3750
3755
3760
3765
3770
3775
3780
3785
3790
3795
3800
3805
3810
3815
3820
3825
3830
3835
3840
3845
3850
3855
3860
3865
3870
3875
3880
3885
3890
3895
3900
3905
3910
3915
3920
3925
3930
3935
3940
3945
3950
3955
3960
3965
3970
3975
3980
3985
3990
3995
4000
4005
4010
4015
4020
4025
4030
4035
4040
4045
4050
4055
4060
4065
4070
4075
4080
4085
4090
4095
4100
4105
4110
4115
4120
4125
4130
4135
4140
4145
4150
4155
4160
4165
4170
4175
4180
4185
4190
4195
4200
4205
4210
4215
4220
4225
4230
4235
4240
4245
4250
4255
4260
4265
4270
4275
4280
4285
4290
4295
4300
4305
4310
4315
4320
4325
4330
4335
4340
4345
4350
4355
4360
4365
4370
4375
4380
4385
4390
4395
4400
4405
4410
4415
4420
4425
4430
4435
4440
4445
4450
4455
4460
4465
4470
4475
4480
4485
4490
4495
4500
4505
4510
4515
4520
4525
4530
4535
4540
4545
4550
4555
4560
4565
4570
4575
4580
4585
4590
4595
4600
4605
4610
4615
4620
4625
4630
4635
4640
4645
4650
4655
4660
4665
4670
4675
4680
4685
4690
4695
4700
4705
4710
4715
4720
4725
4730
4735
4740
4745
4750
4755
4760
4765
4770
4775
4780
4785
4790
4795
4800
4805
4810
4815
4820
4825
4830
4835
4840
4845
4850
4855
4860
4865
4870
4875
4880
4885
4890
4895
4900
4905
4910
4915
4920
4925
4930
4935
4940
4945
4950
4955
4960
4965
4970
4975
4980
4985
4990
4995
5000
5005
5010
5015
5020
5025
5030
5035
5040
5045
5050
5055
5060
5065
5070
5075
5080
5085
5090
5095
5100
5105
5110
5115
5120
5125
5130
5135
5140
5145
5150
5155
5160
5165
5170
5175
5180
5185
5190
5195
5200
5205
5210
5215
5220
5225
5230
5235
5240
5245
5250
5255
5260
5265
5270
5275
5280
5285
5290
5295
5300
5305
5310
5315
5320
5325
5330
5335
5340
5345
5350
5355
5360
5365
5370
5375
5380
5385
5390
5395
5400
5405
5410
5415
5420
5425
5430
5435
5440
5445
5450
5455
5460
5465
5470
5475
5480
5485
5490
5495
5500
5505
5510
5515
5520
5525
5530
5535
5540
5545
5550
5555
5560
5565
5570
5575
5580
5585
5590
5595
5600
5605
5610
5615
5620
5625
5630
5635
5640
5645
5650
5655
5660
5665
5670
5675
5680
5685
5690
5695
5700
5705
5710
5715
5720
5725
5730
5735
5740
5745
5750
5755
5760
5765
5770
5775
5780
5785
5790
5795
5800
5805
5810
5815
5820
5825
5830
5835
5840
5845
5850
5855
5860
5865
5870
5875
5880
5885
5890
5895
5900
5905
5910
5915
5920
5925
5930
5935
5940
5945
5950
5955
5960
5965
5970
5975
5980
5985
5990
5995
6000
6005
6010
6015
6020
6025
6030
6035
6040
6045
6050
6055
6060
6065
6070
6075
6080
6085
6090
6095
6100
6105
6110
6115
6120
6125
6130
6135
6140
6145
6150
6155
6160
6165
6170
6175
6180
6185
6190
6195
6200
6205
6210
6215
6220
6225
6230
6235
6240
6245
6250
6255
6260
6265
6270
6275
6280
6285
6290
6295
6300
6305
6310
6315
6320
6325
6330
6335
6340
6345
6350
6355
6360
6365
6370
6375
6380
6385
6390
6395
6400
6405
6410
6415
6420
6425
6430
6435
6440
6445
6450
6455
6460
6465
6470
6475
6480
6485
6490
6495
6500
6505
6510
6515
6520
6525
6530
6535
6540
6545
6550
6555
6560
6565
6570
6575
6580
6585
6590
6595
6600
6605
6610
6615
6620
6625
6630
6635
6640
6645
6650
6655
6660
6665
6670
6675
6680
6685
6690
6695
6700
6705
6710
6715
6720
6725
6730
6735
6740
6745
6750
6755
6760
6765
6770
6775
6780
6785
6790
6795
6800
6805
6810
6815
6820
6825
6830
6835
6840
6845
6850
6855
6860
6865
6870
6875
6880
6885
6890
6895
6900
6905
6910
6915
6920
6925
6930
6935
6940
6945
6950
6955
6960
6965
6970
6975
6980
6985
6990
6995
7000
7005
7010
7015
7020
7025
7030
7035
7040
7045
7050
7055
7060
7065
7070
7075
7080
7085
7090
7095
7100
7105
7110
7115
7120
7125
7130
7135
7140
7145
7150
7155
7160
7165
7170
7175
7180
7185
7190
7195
7200
7205
7210
7215
7220
7225
7230
7235
7240
7245
7250
7255
7260
7265
7270
7275
7280
7285
7290
7295
7300
7305
7310
7315
7320
7325
7330
7335
7340
7345
7350
7355
7360
7365
7370
7375
7380
7385
7390
7395
7400
7405
7410
7415
7420
7425
7430
7435
7440
7445
7450
7455
7460
7465
7470
7475
7480
7485
7490
7495
7500
7505
7510
7515
7520
7525
7530
7535
7540
7545
7550
7555
7560
7565
7570
7575
7580
7585
7590
7595
7600
7605
7610
7615
7620
7625
7630
7635
7640
7645
7650
7655
7660
7665
7670
7675
7680
7685
7690
7695
7700
7705
7710
7715
7720
7725
7730
7735
7740
7745
7750
7755
7760
7765
7770
7775
7780
7785
7790
7795
7800
7805
7810
7815
7820
7825
7830
7835
7840
7845
7850
7855
7860
7865
7870
7875
7880
7885
7890
7895
7900
7905
7910
7915
7920
7925
7930
7935
7940
7945
7950
7955
7960
7965
7970
7975
7980
7985
7990
7995
8000
8005
8010
8015
8020
8025
8030
8035
8040
8045
8050
8055
8060
8065
8070
8075
8080
8085
8090
8095
8100
8105
8110
8115
8120
8125
8130
8135
8140
8145
8150
8155
8160
8165
8170
8175
8180
8185
8190
8195
8200
8205
8210
8215
8220
8225
8230
8235
8240
8245
8250
8255
8260
8265
8270
8275
8280
8285
8290
8295
8300
8305
8310
8315
8320
8325
8330
8335
8340
8345
8350
8355
8360
8365
8370
8375
8380
8385
8390
8395
8400
8405
8410
8415
8420
8425
8430
8435
8440
8445
8450
8455
8460
8465
8470
8475
8480
8485
8490
8495
8500
8505
8510
8515
8520
8525
8530
8535
8540
8545
8550
8555
8560
8565
8570
8575
8580
8585
8590
8595
8600
8605
8610
8615
8620
8625
8630
8635
8640
8645
8650
8655
8660
8665
8670
8675
8680
8685
8690
8695
8700
8705
8710
8715
8720
8725
8730
8735
8740
8745
8750
8755
8760
8765
8770
8775
8780
8785
8790
8795
8800
8805
8810
8815
8820
8825
8830
8835
8840
8845
8850
8855
8860
8865
8870
8875
8880
8885
8890
8895
8900
8905
8910
8915
8920
8925
8930
8935
8940
8945
8950
8955
8960
8965
8970
8975
8980
8985
8990
8995
9000
9005
9010
9015
9020
9025
9030
9035
9040
9045
9050
9055
9060
9065
9070
9075
9080
9085
9090
9095
9100
9105
9110
9115
9120
9125
9130
9135
9140
9145
9150
9155
9160
9165
9170
9175
9180
9185
9190
9195
9200
9205
9210
9215
9220
9225
9230
9235
9240
9245
9250
9255
9260
9265
9270
9275
9280
9285
9290
9295
9300
9305
9310
9315
9320
9325
9330
9335
9340
9345
9350
9355
9360
9365
9370
9375
9380
9385
9390
9395
9400
9405
9410
9415
9420
9425
9430
9435
9440
9445
9450
9455
9460
9465
9470
9475
9480
9485
9490
9495
9500
9505
9510
9515
9520
9525
9530
9535
9540
9545
9550
9555
9560
9565
9570
9575
9580
9585
9590
9595
9600
9605
9610
9615
9620
9625
9630
9635
9640
9645
9650
9655
9660
9665
9670
9675
9680
9685
9690
9695
9700
9705
9710
9715
9720
9725
9730
9735
9740
9745
9750
9755
9760
9765
9770
9775
9780
9785
9790
9795
9800
9805
9810
9815
9820
9825
9830
9835
9840
9845
9850
9855
9860
9865
9870
9875
9880
9885
9890
9895
9900
9905
9910
9915
9920
9925
9930
9935
9940
9945
9950
9955
9960
9965
9970
9975
9980
9985
9990
9995
9999
10000
10005
10010
10015
10020
10025
10030
10035
10040
10045
10050
10055
10060
10065
10070
10075
10080
10085
10090
10095
10100
10105
10110
10115
10120
10125
10130
10135
10140
10145
10150
10155
10160
10165
10170
10175
10180
10185
10190
10195
10200
10205
10210
10215
10220
10225
10230
10235
10240
10245
10250
10255
10260
10265
10270
10275
10280
10285
10290
10295
10300
10305
10310
10315
10320
10325
10330
10335
10340
10345
10350
10355
10360
10365
10370
10375
10380
10385
10390
10395
10400
10405
10410
10415
10420
10425
10430
10435
10440
10445
10450
10455
10460
10465
10470
10475
10480
10485
10490
10495
10500
10505
10510
10515
10520
10525
10530
10535
10540
10545
10550
10555
10560
10565
10570
10575
10580
10585
10590
10595
10600
10605
10610
10615
10620
10625
10630
10635
10640
10645
10650
10655
10660
10665
10670
10675
10680
10685
10690
10695
10700
10705
10710
10715
10720
10725
10730
10735
10740
10745
10750
10755
10760
10765
10770
10775
10780
10785
10790
10795
10800
10805
10810
10815
10820
10825
10830
10835
10840
10845
10850
10855
10860
10865
10870
10875
10880
10885
10890
10895
10900
10905
10910
10915
10920
10925
10930
10935
10940
10945
10950
10955
10960
10965
10970
10975
10980
10985
10990
10995
11000
11005
11010
11015
11020
11025
11030
11035
11040
11045
11050
11055
11060
11065
11070
11075
11080
11085
11090
11095
11100
11105
11110
11115
11120
11125
11130
11135
11140
11145
11150
11155
11160
11165
11170
11175
11180
11185
11190
11195
11200
11205
11210
11215
11220
11225
11230
11235
11240
11245
11250
11255
11260
11265
11270
11275
11280
11285
11290
11295
11300
11305
11310
11315
11320
11325
11330
11335
11340
11345
11350
11355
11360
11365
11370
11375
11380
11385
11390
11395
11400
11405
11410
11415
11420
11425
11430
11435
11440
11445
11450
11455
11460
11465
11470
11475
11480
11485
11490
11495
11500
11505
11510
11515
11520
11525
11530
11535
11540
11545
11550
11555
11560
11565
11570
11575
11580
11585
11590
11595
11600
11605
11610
11615
11620
11625
11630
11635
11640
11645
11650
11655
11660
11665
11670
11675
11680
11685
11690
11695
11700
11705
11710
11715
11720
11725
11730
11735
11740
11745
11750
11755
11760
11765
11770
11775
11780
11785
11790
11795
11800
11805
11810
11815
11820
11825
11830
11835
11840
11845
11850
11855
11860
11865
11870
11875
11880
11885
11890
11895
11900
11905
11910
11915
11920
11925
11930
11935
11940
11945
11950
11955
11960
11965
11970
11975
11980
11985
11990
11995
12000
12005
12010
12015
12020
12025
12030
12035
12040
12045
12050
12055
12060
12065
12070
12075
12080
12085
12090
12095
12100
12105
12110
12115
12120
12125
12130
12135
12140
12145
12150
12155
12160
12165
12170
12175
12180
12185
12190
12195
12200
12205
12210
12215
12220
12225
12230
12235
12240
12245
12250
12255
12260
12265
12270
12275
12280
12285
12290
12295
12300
12305
12310
12315
12320
12325
12330
12335
12340
12345
12350
12355
12360
12365
12370
12375
12380
12385
12390
12395
12400
12405
12410
12415
12420
12425
12430
12435
12440
12445
12450
12455
12460
12465
12470
12475
12480
12485
12490
12495
12500
12505
12510
12515
12520
12525
12530
12535
12540
12545
12550
12555
12560
12565
12570
12575
12580
12585
12590
12595
12600
12605
12610
12615
12620
12625
12630
12635
12640
12645
12650
12655
12660
12665
12670
12675
12680
12685
12690
12695
12700
12705
12710
12715
12720
12725
12730
12735
12740
12745
12750
12755
12760
12765
12770
12775
12780
12785
12790
12795
12800
12805
12810
12815
12820
12825
12830
12835
12840
12845
12850
12855
12860
12865
12870
12875
12880
12885

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

    theparser.getEsterelModule(theparser.getactualEsterelModule())
        .addStatementToModule(
            theparser.getAddAt(),
            theparser.createStatement((Element) (anElementList.get(i))));

    } // for

    /**
     * Implements the abstract method from <code>EsterelStatement</code>.
     * Returns a string representation of an esterel statement.
     */
    @Override
    public String toString() {
        * none
        * @param anEsterelModule an esterel module representation
        * @return a String representation of an esterel statement
        * @see kiel.fileInterface.estrel.estrel2estudio.EsterelStatement
        * @see kiel.fileInterface.estrel.estrel2estudio.EsterelModule
        */
        public final String toString(final EsterelModule anEsterelModule) {
            210
            String giveback = new String("");
            for (int i = 0; i < this.aSequence.length - 1; i++) {
                220
                giveback
                    += (
                        (EsterelStatement) anEsterelModule.getEsterelProgram().get(
                            this.aSequence[i])).toString(
                                anEsterelModule)
                    + ";";
            }
            return giveback
        }
    }

    /**
     * Fills the substatements in the esterel module.
     */
    @Override
    public void setSubStatements(final EsterelParser theparser)
        throws EsterelParserException {
        170
        * Changes class variable <code>aStatement</code>
        * @param anEsterelStatement an esterel statement
        * @param i the index of the <code>EsterelStatement</code> in
        * <code>aStatement</code>
        * @see kiel.fileInterface.estrel.estrel2estudio.EsterelModule
        */
        public final void setSequenceStatement(final int iStatement, final int i) {
            180
            this.aSequence[i] = aStatement;
        }
    }

    /**
     * Sets the index number from <code>EsterelModule</code>
     * of the substatement.
     */
    @Override
    public void parseEsterelStatement(EsterelParser theparser) {
        190
        * Changes class variable <code>aStatement</code>
        * @param anEsterelStatement an esterel statement
        * @param i the index of the <code>EsterelStatement</code> in
        * <code>aStatement</code>
        * @see kiel.fileInterface.estrel.estrel2estudio.EsterelModule
        */
        public final void setSequenceStatement(final int iStatement, final int i) {
            200
            this.aSequence[i] = aStatement;
        }
    }

    /**
     * Fills the substatements in the esterel module.
     */
    @Override
    public void setSubStatements(final EsterelParser theparser)
        throws EsterelParserException {
        210
        * Changes class variable <code>aStatement</code>
        * @param anEsterelStatement an esterel statement
        * @param i the index of the <code>EsterelStatement</code> in
        * <code>aStatement</code>
        * @see kiel.fileInterface.estrel.estrel2estudio.EsterelModule
        */
        public final void setSequenceStatement(final int iStatement, final int i) {
            220
            this.aSequence[i] = aStatement;
        }
    }

    /**
     * Changes class variable <code>aStatement</code>
     */
    @Override
    public void parseEsterelStatement(EsterelParser theparser) {
        230
        * Changes class variable <code>aStatement</code>
        * @param anEsterelStatement an esterel statement
        * @param i the index of the <code>EsterelStatement</code> in
        * <code>aStatement</code>
        * @see kiel.fileInterface.estrel.estrel2estudio.EsterelModule
        */
        public final void setSequenceStatement(final int iStatement, final int i) {
            240
            this.aSequence[i] = aStatement;
        }
    }
}

```

C.2.41. SuspendEstrelStatement

Der Klassenaufbau ist in der Abbildung C.2.41 dargestellt.

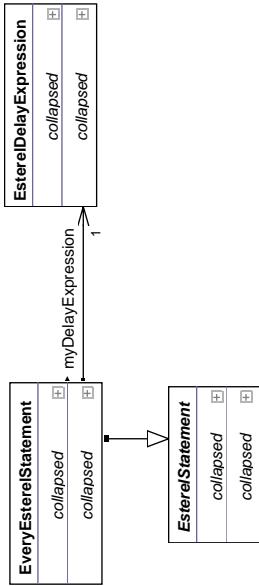


Abbildung C.33.: Klassendiagramm SuspendEstrelStatement

Auflistung C.46: Die Klasse SuspendEstrelStatement

```

/*
 * package kiel.fileInterface.estrel.estrel2estudio;
 */
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.Suspension;
import kiel.fileInterface.estrel.EstrelParser;
import kiel.fileInterface.estrel.EstrelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StatechartHelpers;
import org.json.Element;
*/
/**
 * <p> This is a subclass of <code>EstrelStatement</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 SuspendEstrelStatement extends EstrelStatement {
    /**
     * The DelayExpression.
     */
    private EstrelDelayExpression aDelayExpression;
    /**
     * The type of the expression/
     */
    //private String aDelayExpressionType = "";
    /**
     * The index number of the substitution in an
     * <code>EstrelModule</code>.
     */
    @see kiel.fileInterface.estrel.estrel2estudio.EstrelModule
    /*
     */
    private int aStatement;
    /**
     * Simple constructor.
     */
    public SuspendEstrelStatement() {
        super();
        this.aDelayExpression = null;
    }
    /**
     * Parses a Document and set the class variables.
     */
    public void parseEstrelStatement() {
        DOMParser parser = new DOMParser();
        parser.parse("src/test/resources/estrel/test.s");
        Element root = parser.getDocument();
        Node node = root.getFirstChild();
        while (node != null) {
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                if (node.getTagName().equals("SuspendStatement")) {
                    SuspendStatement ss = new SuspendStatement();
                    ss.parse(node);
                    addStatement(ss);
                }
            }
            node = node.getNextSibling();
        }
    }
}

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

    * @throws EsterelParserException is only delivered
    * @see kiel.file.Interface esterel.EsterelParserException
    * @see #parseEsterelStatement(EsterelParser)
    * @see EsterelStatement#parseEsterelStatement(EsterelParser)
    * @see kiel.file.Interface esterel.EsterelParser
    */
    public SuspendEsterelStatement final EsterelParser theparser)
        throws EsterelParserException {
        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
         * @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 theTreeTrapSignals
         * Stack with all tree trap signals.
         * @throws Esterel2StudioException
         * if the conversion to Kiel is not working. <br>
         * @return a State
         * @see kiel.fileInterface.esterel.esterel2studio.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 theTreeTrapSignals,
            throws Esterel2StudioException {
            State theStatementState =
            (EsterelStatement)
                (anEsterelModule.getEsterelProgram().get(
                    this.aStatement))).convertTokiel(
                    anEsterelModule,
                    true,
                    theLocalEvents,
                    theLocalVariables,
                    theTreeTrapSignals);
            // true means, that the substate is final
            Suspend aSuspend = new Suspend();
            Suspension theSuspensionTransition = new Suspension();
            StateChartHelpers.set(
                theSuspensionTransition,
                aSuspend,
                0,
                theStatementState);
            theSuspensionTransition.setLabel(
                this.aDelayExpression.convertTokiel(
                    anEsterelModule,
                    theLocalEvents,
                    theLocalVariables,
                    null));
            InitialState anInitialState = new InitialState();
            InitialArc aInitialArc = new InitialArc();
            aInitialArc.setSource(anInitialState);
            aInitialArc.setTarget(theStatementState);
            if (isFinalState) {
                return StateChartHelpers.createFinalOR(
                    this.getEsterelStatementName() + "state",
                    new Node[] { anInitialState, aSuspend, theStatementState });
            }
            /**
             * Implements the abstract methode from <code>EsterelStatement</code>
             * Parses an esterel state in an <code>jdom.Document</code>
             * representation of an .exp file.
             * if you have not called the constructor
             * use the methode <code>parseEsterelStatement</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#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.get XPathSearchString());
                if (anElementList.size() == 2) {
                    theparser.incStatementCounter();
                    this.aDelayExpression =
                        new EsterelDelayExpression(
                            theparser,
                            this.get XPathSearchString()
                                + "[local-name() = "
                                + ((Element) anElementList.get(1)).getName()
                                + ", /* ] *",
                                theparser.get XMLDocument());
                }
            }
        }
    }
}

```

```

    } //public final void parseEstrelStatement(final EstrelParser theparser) {
    /**
     * Fills the substatements in the estrel module
     * @param Sideffects: changes EstrelModule.
     */
    public final void setSuspendStatement(final int aStatementIndex) {
        this.aStatement = aStatementIndex;
    }
    /**
     * Implements the abstract methode from <code>EstrelStatement</code>
     * Returns a string representation of an estrel statement.
     */
    public final String toString(final EstrelModule anEstrelModule) {
        return this.getEstrelStatementName()
    }
    /**
     * @throws EstrelParserException is only delivered
     * @see kiel.fileInterface.estrel.EstrelParserException
     */
    public final void setSubStatements(final EstrelParser theparser)
        throws EstrelParserException {
        List anElementList = new ArrayList();
        DOMHelpers.getElements(
            theparser.getXMLDocument(),
            this.getXPathSearchString());
        this.setSuspendStatement(theparser.getAddAt());
        theparser.getEstrelModule(theparser.getActualEstrelModule())
            .addStatementToModule(
                theparser.getAddAt(),
                theparser.createStatement(((Element) (anElementList.get(0)))));
    }
    /**
     * Sets the index number from <code>EstrelModule</code>
     * of the substatement.
     */
    public final void setAddAt(int aStatementIndex) {
        this.aStatement = aStatementIndex;
    }
    /**
     * Changes class variable <code>aStatement</code>
     * @param aStatementIndex an index
     */
    public final void setSuspendStatement(int aStatementIndex) {
        this.aStatement = aStatementIndex;
    }
}

```

C.2.42. SustainEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.42 dargestellt.

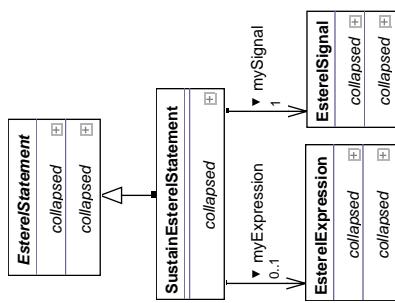


Abbildung C.34.: Klassendiagramm SustainEsterelStatement

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.CompoundLabelParser;
import kiel.util.CompoundLabelParser;
import kiel.util.DONHelpers;
import kiel.util.StateCharChangers;
import org.json.Element;
  
```

20

```

    /**
     * <p>
     * This is a subclass of <code>EsterelStatement</code>
     * classes implements the sustain statement.
     * </p>
     * <p>
     * It implements await s, await s do p; await case s do p; await case s
     * </p>
     * <p>
     * Copyright: Copyright (c) 2005
     * </p>
     * <p>
     * </p>
     * <p>
     * Company: Uni Kiel
     * </p>
     * <p>
     * @author <a href="mailto:lkuehnformatik.uni-kiel.de">Lars Kuehl </a>
     * <br>
     * <br>
     * </p>
    */
  
```

40

```

public class SustainEsterelStatement {
    extends EsterelStatement {
        /**
         * The expression of the signal to emit.
         */
  
```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

private EsterelExpression myExpression;
    * @throws Esterel2EstudioException
    * if the conversion to Kiel is not working. <br>
    */
    /**
     * The signal to emit. */
    private EsterelSignal mySignal = null;

    /**
     * simple constructor.
     */
    public SustainEsterelStatement() {
        super();
        this.myExpression = null;
    }

    /**
     * Parses a Document and set the class variables. <br>
     * It calls also the super constructor <code>EsterelStatement</code>.
     * (EsterelParser theparser)</code>.
     * Calls the <code>parseEsterelStatement</code> method <br>
     * Sideeffects: Changes <code>EsterelParser</code> class variables
     * @param theparser
     * an intace of a EsterelParser
     * @throws EsterelParserException
     * is only delivered
     * @see kiel.fileInterface.estrel.EsterelParserException
     * @see #parseEsterelStatement(EsterelParser)
     * @see EsterelStatement#esterelStatement(EsterelParser)
     * @see kiel.fileInterface.estrel.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 estrel statement.
     * @param amEsterelModule
     * @param theLocalEvents
     * is true if the new state has to be a final state
     * Stack with the local events
     * @param theLocalVariables
     * Stack with all local variables
     * @param theTreesTrapSignals
     * Stack with all tree trap signals.
     */
    public 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.createSs(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()
            + ")";
        }
        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 });
         */
    }
}

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

    */
    return StateChartHelpers.createOR(this.getEsterelStatementName()
        + "state",
        new Node[] {
            anInitialState, theStartState });
    }
    /**
     * @param final kiel.dataStructure.State convertToKiel
    */

    /**
     * Implements the abstract methode from <code>EsterelStatement</code>
     * Parses an esterel statement 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.estrel.EsterelParserException
     * @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see EsterelStatement#parseEsterelStatement(EsterelParser)
     * @see EsterelStatement#parseEsterelStatement(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.getPathSearchString());
        if (anElementList.size() > 0
            && anElementList.size() < toomuch) {
            this.mySignal =
                theparser.getEsterelModule(theparser.getActualEsterelModule())
                    .getEsterelSignalByID(this.getPathSearchString());
            if (anElementList.size() > 1
                && anElementList.size() < toomuch) {
                final int theIndex = 1;
                this.myExpression = new EsterelExpression(
                    theparser, this.getPathSearchString()
                        + "[local-name(.)=""
                        + ((Element) anElementList.get(theIndex))
                            .getName()
                        + ",]@id=""
                        + ((Element) anElementList.get(theIndex))
                            .getAttributeValue("id")
                        + ",]*", theparser.getXMLDocument());
            }
        }
    }

    /**
     * else {
     *     throw new EsterelParserException(
     *         this.getEsterelStatementName()
     *             + " no signal expression", theparser);
     * } //if >
    */
    //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.estrel.EsterelParserException
     * @see kiel.fileInterface.estrel.EsterelParser
    */
    public final void setSubStatements(
        final EsterelParser theparser)
        throws EsterelParserException {
    }

    /**
     * Implements the abstract 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.estrel.EsterelStatement
     * @see kiel.fileInterface.estrel.EsterelModule
    */
    public final String toString(
        final EsterelModule anEsterelModule) {
        return this.getEsterelStatementName()
            + " ";
    }

    /**
     * Returns the signal of the SignalExpression. <br>
     * Sideeffects: none
     * @param anElementList
     *      the children elements of the SignalExpression
     * @return the signals id
     * @throws EsterelParserException
     *      if no signal is found
     */
    private String getSignal(
        final List anElementList)
    {
        210
        220
        230
        240
        250
        260
    }
}

```

```
throws EstrelParserException {
    final int theSignalIndex = 0;
    if (anElementList.size() > theSignalIndex) {
        return ((Element) anElementList.get(theSignalIndex))
            .getAttributeValue("id");
    }
}
```

270

283

C.2.43. TaskCallEsterelStatement

Der Klassenaufbau ist in der Abbildung C.2.43 dargestellt.

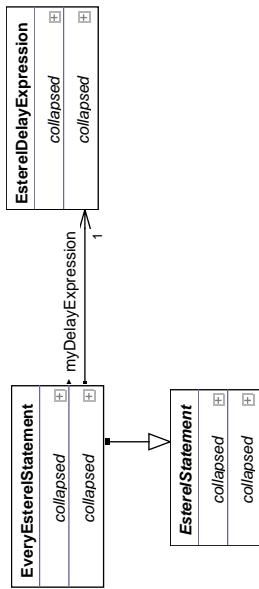


Abbildung C.35.: Klassendiagramm TaskCallEsterelStatement

Aufzählung C.48: Die Klasse TaskCallEstimateStatement

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

    /**
     * The value parameters.
     */
    private EstrelExpression[] myValueParameter = null;
    /**
     * the return signal.
     */
    private EstrelSignal theReturnSignal = null;

    /**
     * Simple constructor.
     */
    public TaskCallEstrelStatement() {
        super();
        final int m = -2;
        this.myStatement = m;
    }

    /**
     * Parses a Document and set the class variables. <br>
     * It calls also the super constructor <code>EstrelStatement</code>
     * (EstrelParser theparser)</code>.
     * <br>
     * Calls the <code>parseEstrelStatement</code> methode <br>
     * Sideffects: Changes <code>EstrelParser</code> class variables
     *
     * @param theparser
     * an intace of a EstrelParser
     * @throws EstrelParserException aXception
     * @see #parseEstrelStatement(EstrelParser)
     * @see EstrelStatement#parseEstrelStatement(EstrelParser)
     */
    public TaskCallEstrelStatement(final EstrelParser theparser)
        throws EstrelParserException {
        super(theparser);
        final int m = -2;
        this.myStatement = m;
        this.parseEstrelStatement(theparser);
    }

    /**
     * Implements the methode from <code>EstrelStatement</code>.
     * Returns a state representation of the estrel statement
     *
     * @param anEstrelModule
     * @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
     */
    public final kiel.dataStructure.State convertToKiel(
        final EstrelModule anEstrelModule,
        final boolean isFinalState,
        final Arraylist<theLocalEvents,
        final Arraylist<theLocalVariables,
        final Arraylist<theTreesTrapsignals>
            throws Estrel2EstudioException {
        if (this.myStatement > -1) {
            return ((EstrelStatement) anEstrelModule).convertToKiel(anEstrelModule,
                .get(this.myStatement)).convertToKiel(anEstrelModule,
                true,
                theLocalEvents,
                theLocalVariables,
                theTreesTrapsignals);
        }
        InitialArc aInitialArc = new InitialArc();
        InitialArc anInitialArc = new InitialArc();
        FinalSimplestate theEndState = new FinalSimplestate();
        anInitialArc.setSource(anInitialState);
        anInitialArc.setTarget(theEndState);
        return StateChartHelpers.createInitialOR(this.getEstrelStatementName()
            + " state",
            new Note[] {
                anInitialState, theEndState });
    }

    /**
     * @return Returns the theReturnSignalID.
     */
    public final EstrelSignal getTheReturnSignalID() {
        return this.theReturnSignal;
    }

    /**
     * Implements the abstract methode from <code>EstrelStatement</code>
     * Parses an estrel statment in an <code>org.jdom.Document</code>
     * representation of an exp file if you have not called the
     * constructor <code>EstrelStatement(EstrelParser theparser)</code>
     * use the methode <code>reparseEstrelStatement</code> before
     * <code>parseEstrelStatement</code> In this special case Taskcall
     * has to be done. <br>
     * Sideffects: none Changes <code>EstrelParser</code> class
     * variables
     *
     * @param theparser
     * an instance of <code>EstrelParser</code>
     */
    public void EstrelParserException if something goes wrong.
    * @see EstrelStatement#EstrelStatement(EstrelParser theparser)

```

```

  * @see EsterelStatement#parseEsterelStatement(EsterelParser)
  * @see kiel.fileInterface.esterel.EsterelParser
  */
  public final void parseEsterelStatement(
    final EsterelParser theparser)
    throws EsterelParserException {
    final int ATRIE = 3;
    final int aFOUR = 4;
    List anElementList = DOMHelpers.getElements(this.get XPathSearchString(),
      theparser.getXMLDocument());
    int theEDVCounter = 0;
    int i = 0;
    ArrayList theReferenceParameter = new ArrayList();
    ArrayList theValueParameter = new ArrayList();
    while (theEDVCounter < aFOUR) {
      && i < anElementList.size() {
        Element anElement = (Element) anElementList.get(i);
        if (anElement.getAttribute("name").compareTo("EOV") == 0) {
          theEDVCounter++;
        } else if (i == 0) { // parse name
          this.myTask =
            theparser
              .getEsterelModule(theparser.getActualEsterelModule())
              .getEsterelTaskByID(anElement.getAttributeValue("id"));
        } else if (theEDVCounter == 0) {
          theReferenceCounter.add(
            theparser
              .getEsterelModule(theparser.getActualEsterelModule())
              .getEsterelVariableByID(anElement.getAttributeValue("id")));
        } else if (theEDVCounter == 1) {
          theValueParameter.add(new EsterelExpression(
            theparser,
            "/*[@id=" + anElement.getAttributeValue("id")
              + "]/*/",
            theparser.getXMLDocument()));
        }
      }
    }
    if (theEDVCounter == 2) {
      this.theReturnSignal =
        theparser
          .getEsterelModule(theparser.getActualEsterelModule())
          .getEsterelSignalByID(anElement.getAttributeValue("id")));
    }
    theEDVCounter++;
  }
  /**
   * Implements the abstrac methode from <code>EsterelStatement</code>.
   * Returns a string representation of an esterel statement
   * @param anEsterelModule
   * @return an esterel modul representation.
   */
  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; i++) {
        if (int i = 0; j < theReferenceParameter.size(); j++) {
          this.myReferenceParameter[j] =
            theReferenceParameter[i].getRefereceParameter();
        }
      }
    }
    return result;
  }
}

```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

    + EsterelEstudioProperties.getSeparatorString();
}
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);
    }
    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.

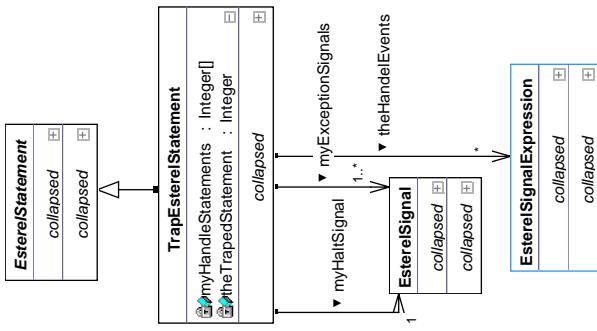


Abbildung C.36.: Klassendiagramm TrapEsterelStatement

Auflistung C.49: Die Klasse TrapEsterelStatement

```

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

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

import kiel.dataStructure.State;
import kiel.dataStructure.StringLabel;
import kiel.dataStructure.Transition;
import kiel.dataStructure.TransitionLabel;
import kiel.dataStructure.WeakAbortion;
import kiel.dataStructure.BooleanExp BooleanExpression;
import kiel.dataStructure.eventExp.BooleanOr;
import kiel.dataStructure.eventExp.DelayExpression;
import kiel.dataStructure.eventExp.Event;
import kiel.dataStructure.estrel.Parser;
import kiel.fileInterface.estrel.EsterelParserException;
import kiel.util.DOMHelpers;
import kiel.util.StateChartHelpers;
import kiel.util.StateUtil;

import org.json.Element;
import org.json.JSONObject;
import org.json.XML;

private int theTrapedStatement;
/** 
 * A new local signal which is emitted when the trap has to hold.
 */
private EsterelSignal myHaltSignal = null;
/** 
 * Simple constructor.
 */
public TrapEsterelStatement() {
    super();
    theHandleEvents = null;
    theTrapedStatement = -1;
    myExceptionSignals = null;
    myHandleStatements = null;
}

/**
 * Parses a Document and set the class variables. <br>
 * It calls also the super constructor <code>EsterelStatement</code>
 * (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
*
* @see kiel.fileInterface.estrel.EsterelParserException
* @see #parseEsterelStatement(EsterelParser)
* @see EsterelStatement#EsterelStatement(EsterelParser)
* @see kiel.fileInterface.estrel.EsterelParser
*/
public TrapEsterelStatement(final EsterelParser theparser)
throws EsterelParserException {
    super(
        theparser);
    theHandleEvents = null;
    theTrapedStatement = -1;
    myExceptionSignals = null;
    myHandleStatements = null;
    myHaltSignal = theparser.getEsterelModule(
        theparser.getActualEsterelModule()
            .addNewSignal());
    this.myHaltSignal.setTrap(true);
    this.myHaltSignal.setMySignalIdentifier("halt");
    this.myHaltSignal.createEsterelStatementName();
    this.myHaltSignal.createEsterelIdentifierName();
    theparser.getEsterelModule(theparser.getActualEsterelModule());
}

/**
 * Implements the methode from <code>EsterelStatement</code>.
 */
// Public TrapEsterelStatement(final EsterelParser theparser) {
}

/*
 * Implements the state representation of the estrel statement
 */
// Returns a state representation of the estrel statement
}

/*
 * <br>
 * This is a subclass of <code>EsterelStatement</code>
 */
classes.
</p>
<p>
* Copyright : Copyright (c) 2004
* </p>
<p>
* Company: Uni Kiel
* @author <a href="mailto:luke@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;
60
/**
 * true if the ExceptionSignal has no handler. */
private boolean[] pureHandle;

/**
 * the exception events which starts a handel.
 */
private EsterelSignalExpression[] theHandleEvents;
70
/**
 * the index number of the substatement in an
 * <code>esterelModule</code>.
 */
* @see kiel.fileInterface.estrel.EsterelModule
*/

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

*   * @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 theTreeTrapSignals
*   *   Stack with all tree trap signals.
*   * @throws Esterel2StudioException
*   *   if the conversion 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<theTreeTrapSignals>
    throws Esterel2StudioException {
    int priority = 1;
    InitialArc anInitialArc = new InitialArc();
    CompositeState anInitialState = new InitialState();
    FinalSimpleState theNormalTermState = new FinalSimpleState();
    SimpleState theTrapState = new SimpleState();
    State theHandleState = null;
    if (isFinalState) {
        State = new FinalORState(
            this.getEsterelStatementName()
                + " state");
    } else {
        State = 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 statemachine
    StateChartHelpers.addLocalEvents(aState,
        theEvents);
    aState.addLocalEvent(this.myHaltSignal_convertToKiel(anEsterelModule));
    theEvents.size();
    theTreeTrapSignals.add(this);
    theLocalVariables.add(this);
}
StateStatementState = (EsterelStatement) anEsterelModule,
true

```


C. Java Code für die Transformation von Esterel in SyncCharts

```

if (theHandleState != null) {
    theHandleState.setParent(aState);
    aState.addSubnode(theHandleState);
}
theNormalTermState.setParent(aState);
aState.addSubnode(theNormalTermState);
anInitialArc.setSource(anInitialState);
anInitialArc.setTarget(anStatementState);
anInitialState.setParent(aState);
aStatementState.setParent(aState);
aState.addSubnode(aStatementState);
aState.addSubnode(anInitialState);
420
} //we added all the localsignals to theLocalEvents
// now we delete them again
while (theLocalEvents.size() != oldsize) {
    theLocalEvents.remove(theLocalEvents.size() - 1);
}
return aState;
}
430
//public final kiel.datastructure.State convertToKiel(
/*
 * Implements the abstract methode from <code>EsterelStatement</code>
 * Parses an esterel statement 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>parseEsterelStatement</code> before
 * <code>parseEsterelListStatement</code>.<br>
 * Sideeffect: the class variables are set Changes
 * <code>EsterelParser</code> class variables
 *
 * @param theparser
 *      an instance of <code>EsterelParser</code>
 * @throws EsterelParserException
 *      is only delivered
 *      if the fileInterface esterel.EsterelParserException
 *      is seen
 *      EsterelStatement#parseEsterelStatement(EsterelParser theparser)
 *      EsterelStatement#parseEsterelListStatement(EsterelParser theparser)
 *      EsterelFileInterface esterel.EsterelParser
 *      org.jdom.Document
 */
440
public final void parseEsterelStatement (
    final EsterelParser theparser)
    throws EsterelParserException {
    List anElementList = DOMHelpers.getElements(theparser.getDocument(),
        this.getPathAsString());
    int anElementIndex = 0;
    if (anElementList.size() > 0
        && ((Element) anElementList.get(anElementIndex)).getName()
            .compareTo("NULL") == 0) {
        throw new EsterelParserException(
            this.getEsterelStatementName()
                + " no Statement found");
    }
    360
    if (anElementList.size() > 1
        && ((Element) anElementList.get(anElementIndex)).getName()
            .compareTo("NULL") == 0) {
        throw new EsterelParserException(
            this.getEsterelStatementName()
                + " no exception declaration found ");
    }
    370
    List anExceptionDeclarationReferenceList =
        DOMHelpers.getElements(theparser.getDocument(),
            /*@ [id=""
            + ((Element) anElementList.get(anElementIndex))
            .getAttributeValue("id")
            + "?"]/*not(local-name(.)='EOF')*/
            + "[not(local-name(.)='SymbolTable')]\"");
    380
    390
    400
    410
    420
    430
    440
    450
    460
    470
    480
    490
    500
    510
    520
    530
    540
    550
    560
    570
    580
    590
    600
    610
    620
    630
    640
    650
    660
    670
    680
    690
    700
    710
    720
    730
    740
    750
    760
    770
    780
    790
    800
    810
    820
    830
    840
    850
    860
    870
    880
    890
    900
    910
    920
    930
    940
    950
    960
    970
    980
    990
    1000
    1010
    1020
    1030
    1040
    1050
    1060
    1070
    1080
    1090
    1100
    1110
    1120
    1130
    1140
    1150
    1160
    1170
    1180
    1190
    1200
    1210
    1220
    1230
    1240
    1250
    1260
    1270
    1280
    1290
    1300
    1310
    1320
    1330
    1340
    1350
    1360
    1370
    1380
    1390
    1400
    1410
    1420
    1430
    1440
    1450
    1460
    1470
    1480
    1490
    1500
    1510
    1520
    1530
    1540
    1550
    1560
    1570
    1580
    1590
    1600
    1610
    1620
    1630
    1640
    1650
    1660
    1670
    1680
    1690
    1700
    1710
    1720
    1730
    1740
    1750
    1760
    1770
    1780
    1790
    1800
    1810
    1820
    1830
    1840
    1850
    1860
    1870
    1880
    1890
    1900
    1910
    1920
    1930
    1940
    1950
    1960
    1970
    1980
    1990
    2000
    2010
    2020
    2030
    2040
    2050
    2060
    2070
    2080
    2090
    2100
    2110
    2120
    2130
    2140
    2150
    2160
    2170
    2180
    2190
    2200
    2210
    2220
    2230
    2240
    2250
    2260
    2270
    2280
    2290
    2300
    2310
    2320
    2330
    2340
    2350
    2360
    2370
    2380
    2390
    2400
    2410
    2420
    2430
    2440
    2450
    2460
    2470
    2480
    2490
    2500
    2510
    2520
    2530
    2540
    2550
    2560
    2570
    2580
    2590
    2600
    2610
    2620
    2630
    2640
    2650
    2660
    2670
    2680
    2690
    2700
    2710
    2720
    2730
    2740
    2750
    2760
    2770
    2780
    2790
    2800
    2810
    2820
    2830
    2840
    2850
    2860
    2870
    2880
    2890
    2900
    2910
    2920
    2930
    2940
    2950
    2960
    2970
    2980
    2990
    3000
    3010
    3020
    3030
    3040
    3050
    3060
    3070
    3080
    3090
    3100
    3110
    3120
    3130
    3140
    3150
    3160
    3170
    3180
    3190
    3200
    3210
    3220
    3230
    3240
    3250
    3260
    3270
    3280
    3290
    3300
    3310
    3320
    3330
    3340
    3350
    3360
    3370
    3380
    3390
    3400
    3410
    3420
    3430
    3440
    3450
    3460
    3470
    3480
    3490
    3500
    3510
    3520
    3530
    3540
    3550
    3560
    3570
    3580
    3590
    3600
    3610
    3620
    3630
    3640
    3650
    3660
    3670
    3680
    3690
    3700
    3710
    3720
    3730
    3740
    3750
    3760
    3770
    3780
    3790
    3800
    3810
    3820
    3830
    3840
    3850
    3860
    3870
    3880
    3890
    3900
    3910
    3920
    3930
    3940
    3950
    3960
    3970
    3980
    3990
    4000
    4010
    4020
    4030
    4040
    4050
    4060
    4070
    4080
    4090
    4100
    4110
    4120
    4130
    4140
    4150
    4160
    4170
    4180
    4190
    4200
    4210
    4220
    4230
    4240
    4250
    4260
    4270
    4280
    4290
    4300
    4310
    4320
    4330
    4340
    4350
    4360
    4370
    4380
    4390
    4400
    4410
    4420
    4430
    4440
    4450
    4460
    4470
    4480
    4490
    4500
    4510
    4520
    4530
    4540
    4550
    4560
    4570
    4580
    4590
    4600
    4610
    4620
    4630
    4640
    4650
    4660
    4670
    4680
    4690
    4700
    4710
    4720
    4730
    4740
    4750
    4760
    4770
    4780
    4790
    4800
    4810
    4820
    4830
    4840
    4850
    4860
    4870
    4880
    4890
    4900
    4910
    4920
    4930
    4940
    4950
    4960
    4970
    4980
    4990
    5000
    5010
    5020
    5030
    5040
    5050
    5060
    5070
    5080
    5090
    5100
    5110
    5120
    5130
    5140
    5150
    5160
    5170
    5180
    5190
    5200
    5210
    5220
    5230
    5240
    5250
    5260
    5270
    5280
    5290
    5300
    5310
    5320
    5330
    5340
    5350
    5360
    5370
    5380
    5390
    5400
    5410
    5420
    5430
    5440
    5450
    5460
    5470
    5480
    5490
    5500
    5510
    5520
    5530
    5540
    5550
    5560
    5570
    5580
    5590
    5600
    5610
    5620
    5630
    5640
    5650
    5660
    5670
    5680
    5690
    5700
    5710
    5720
    5730
    5740
    5750
    5760
    5770
    5780
    5790
    5800
    5810
    5820
    5830
    5840
    5850
    5860
    5870
    5880
    5890
    5900
    5910
    5920
    5930
    5940
    5950
    5960
    5970
    5980
    5990
    6000
    6010
    6020
    6030
    6040
    6050
    6060
    6070
    6080
    6090
    6100
    6110
    6120
    6130
    6140
    6150
    6160
    6170
    6180
    6190
    6200
    6210
    6220
    6230
    6240
    6250
    6260
    6270
    6280
    6290
    6300
    6310
    6320
    6330
    6340
    6350
    6360
    6370
    6380
    6390
    6400
    6410
    6420
    6430
    6440
    6450
    6460
    6470
    6480
    6490
    6500
    6510
    6520
    6530
    6540
    6550
    6560
    6570
    6580
    6590
    6600
    6610
    6620
    6630
    6640
    6650
    6660
    6670
    6680
    6690
    6700
    6710
    6720
    6730
    6740
    6750
    6760
    6770
    6780
    6790
    6800
    6810
    6820
    6830
    6840
    6850
    6860
    6870
    6880
    6890
    6900
    6910
    6920
    6930
    6940
    6950
    6960
    6970
    6980
    6990
    7000
    7010
    7020
    7030
    7040
    7050
    7060
    7070
    7080
    7090
    7100
    7110
    7120
    7130
    7140
    7150
    7160
    7170
    7180
    7190
    7200
    7210
    7220
    7230
    7240
    7250
    7260
    7270
    7280
    7290
    7300
    7310
    7320
    7330
    7340
    7350
    7360
    7370
    7380
    7390
    7400
    7410
    7420
    7430
    7440
    7450
    7460
    7470
    7480
    7490
    7500
    7510
    7520
    7530
    7540
    7550
    7560
    7570
    7580
    7590
    7600
    7610
    7620
    7630
    7640
    7650
    7660
    7670
    7680
    7690
    7700
    7710
    7720
    7730
    7740
    7750
    7760
    7770
    7780
    7790
    7800
    7810
    7820
    7830
    7840
    7850
    7860
    7870
    7880
    7890
    7900
    7910
    7920
    7930
    7940
    7950
    7960
    7970
    7980
    7990
    8000
    8010
    8020
    8030
    8040
    8050
    8060
    8070
    8080
    8090
    8100
    8110
    8120
    8130
    8140
    8150
    8160
    8170
    8180
    8190
    8200
    8210
    8220
    8230
    8240
    8250
    8260
    8270
    8280
    8290
    8300
    8310
    8320
    8330
    8340
    8350
    8360
    8370
    8380
    8390
    8400
    8410
    8420
    8430
    8440
    8450
    8460
    8470
    8480
    8490
    8500
    8510
    8520
    8530
    8540
    8550
    8560
    8570
    8580
    8590
    8600
    8610
    8620
    8630
    8640
    8650
    8660
    8670
    8680
    8690
    8700
    8710
    8720
    8730
    8740
    8750
    8760
    8770
    8780
    8790
    8800
    8810
    8820
    8830
    8840
    8850
    8860
    8870
    8880
    8890
    8900
    8910
    8920
    8930
    8940
    8950
    8960
    8970
    8980
    8990
    9000
    9010
    9020
    9030
    9040
    9050
    9060
    9070
    9080
    9090
    9100
    9110
    9120
    9130
    9140
    9150
    9160
    9170
    9180
    9190
    9200
    9210
    9220
    9230
    9240
    9250
    9260
    9270
    9280
    9290
    9300
    9310
    9320
    9330
    9340
    9350
    9360
    9370
    9380
    9390
    9400
    9410
    9420
    9430
    9440
    9450
    9460
    9470
    9480
    9490
    9500
    9510
    9520
    9530
    9540
    9550
    9560
    9570
    9580
    9590
    9600
    9610
    9620
    9630
    9640
    9650
    9660
    9670
    9680
    9690
    9700
    9710
    9720
    9730
    9740
    9750
    9760
    9770
    9780
    9790
    9800
    9810
    9820
    9830
    9840
    9850
    9860
    9870
    9880
    9890
    9900
    9910
    9920
    9930
    9940
    9950
    9960
    9970
    9980
    9990
    10000
    10010
    10020
    10030
    10040
    10050
    10060
    10070
    10080
    10090
    10100
    10110
    10120
    10130
    10140
    10150
    10160
    10170
    10180
    10190
    10200
    10210
    10220
    10230
    10240
    10250
    10260
    10270
    10280
    10290
    10300
    10310
    10320
    10330
    10340
    10350
    10360
    10370
    10380
    10390
    10400
    10410
    10420
    10430
    10440
    10450
    10460
    10470
    10480
    10490
    10500
    10510
    10520
    10530
    10540
    10550
    10560
    10570
    10580
    10590
    10600
    10610
    10620
    10630
    10640
    10650
    10660
    10670
    10680
    10690
    10700
    10710
    10720
    10730
    10740
    10750
    10760
    10770
    10780
    10790
    10800
    10810
    10820
    10830
    10840
    10850
    10860
    10870
    10880
    10890
    10900
    10910
    10920
    10930
    10940
    10950
    10960
    10970
    10980
    10990
    11000
    11010
    11020
    11030
    11040
    11050
    11060
    11070
    11080
    11090
    11100
    11110
    11120
    11130
    11140
    11150
    11160
    11170
    11180
    11190
    11200
    11210
    11220
    11230
    11240
    11250
    11260
    11270
    11280
    11290
    11300
    11310
    11320
    11330
    11340
    11350
    11360
    11370
    11380
    11390
    11400
    11410
    11420
    11430
    11440
    11450
    11460
    11470
    11480
    11490
    11500
    11510
    11520
    11530
    11540
    11550
    11560
    11570
    11580
    11590
    11600
    11610
    11620
    11630
    11640
    11650
    11660
    11670
    11680
    11690
    11700
    11710
    11720
    11730
    11740
    11750
    11760
    11770
    11780
    11790
    11800
    11810
    11820
    11830
    11840
    11850
    11860
    11870
    11880
    11890
    11900
    11910
    11920
    11930
    11940
    11950
    11960
    11970
    11980
    11990
    12000
    12010
    12020
    12030
    12040
    12050
    12060
    12070
    12080
    12090
    12100
    12110
    12120
    12130
    12140
    12150
    12160
    12170
    12180
    12190
    12200
    12210
    12220
    12230
    12240
    12250
    12260
    12270
    12280
    12290
    12300
    12310
    12320
    12330
    12340
    12350
    12360
    12370
    12380
    12390
    12400
    12410
    12420
    12430
    12440
    12450
    12460
    12470
    12480
    12490
    12500
    12510
    12520
    12530
    12540
    12550
    12560
    12570
    12580
    12590
    12600
    12610
    12620
    12630
    12640
    12650
    12660
    12670
    12680
    12690
    12700
    12710
    12720
    12730
    12740
    12750
    12760
    12770
    12780
    12790
    12800
    12810
    12820
    12830
    12840
    12850
    12860
    12870
    12880
    12890
    12900
    12910
    12920
    12930
    12940
    12950
    12960
    12970
    12980
    12990
    13000
    13010
    13020
    13030
    13040
    13050
    13060
    13070
    13080
    13090
    13100
    13110
    13120
    13130
    13140
    13150
    13160
    13170
    13180
    13190
    13200
    13210
    13220
    13230
    13240
    13250
    13260
    13270
    13280
    13290
    13300
    13310
    13320
    13330
    13340
    13350
    13360
    13370
    13380
    13390
    13400
    13410
    13420
    13430
    13440
    13450
    13460
    13470
    13480
    13490
    13500
    13510
    13520
    13530
    13540
    13550
    13560
    13570
    13580
    13590
    13600
    13610
    13620
    13630
    13640
    13650
    13660
    13670
    13680
    13690
    13700
    13710
    13720
    13730
    13740
    13750
    13760
    13770
    13780
    13790
    13800
    13810
    13820
    13830
    13840
    13850
    13860
    13870
    13880
    13890
    13900
    13910
    13920
    13930
    13940
    13950
    13960
    13970
    13980
    13990
    14000
    14010
    14020
    14030
    14040
    14050
    14060
    14070
    14080
    14090
    14100
    14110
    14120
    14130
    14140
    14150
    14160
    14170
    14180
    14190
    14200
    14210
    14220
    14230
    14240
    14250
    14260
    14270
    14280
    14290
    14300
    14310
    14320
    14330
    14340
    14350
    14360
    14370
    14380
    14390
    14400
    14410
    14420
    14430
    14440
    14450
    14460
    14470
    14480
    14490
    14500
    14510
    14520
    14530
    14540
    14550
    14560
    14570
    14580
    14590
    14600
    14610
    14620
    14630
    14640
    14650
    14660
    14670
    14680
    14690
    14700
    14710
    14720
    14730
    14740
    14750
    14760
    14770
    14780
    14790
    14800
    14810
    14820
    14830
    14840
    14850
    14860
    14870
    14880
    14890
    14900
    14910
    14920
    14930
    14940
    14950
    14960
    14970
    14980
    14990
    15000
    15010
    15020
    15030
    15040
    15050
    15060
    15070
    15080
    15090
    15100
    15110
    15120
    15130
    15140
    15150
    15160
    15170
    15180
    15190
    15200
    15210
    15220
    15230
    15240
    15250
    15260
    15270
    15280
    15290
    15300
    15310
    15320
    15330
    15340
    15350
    15360
    15370
    15380
    15390
    15400
    15410
    15420
    15430
    15440
    15450
    15460
    15470
    15480
    15490
    15500
    15510
    15520
    15530
    15540
    15550
    15560
    15570
    15580
    15590
    15600
    15610
    15620
    15630
    15640
    15650
    15660
    15670
    15680
    15690
    15700
    15710
    15720
    15730
    15740
    15750
    15760
    15770
    15780
    15790
    15800
    15810
    15820
    15830
    15840
    15850
    15860
    15870
    15880
    15890
    15900
    15910
    15920
    15930
    15940
    15950
    15960
    15970
    15980
    15990
    16000
    16010
    16020
    16030
    16040
    16050
    16060
    16070
    16080
    16090
    16100
    16110
    16120
    16130
    16140
    16150
    16160
    16170
    1
```

C.2. Paket kiel.fileInterface.estrel.estrel2estudio

```

for (int theHandleIndex = 0;
theHandleIndex < theHandlerList.size();
theHandleIndex++) {
    if (((Element) theHandlerList.get(theHandleIndex)).getTagName()
        .compareTo("PredicatedStatement") == 0) {
        String aPredicatedStatementPathSearchString =
            String aPredicatedStatementPathSearchString =
                this.getXPatchSearchString()
                + "[Local-name(.)=" +
                + ((Element) theHandlerList.get(theHandleIndex))
                .getTagName()
                + "] [@id=" +
                + ((Element) theHandlerList.get(theHandleIndex))
                .getAttributeValue("id")
                + "]*";
        List aPredicatedElementList = DOMHelpers.getElements(
            theParser.getXMLDocument(),
            aPredicatedStatementPathSearchString);
        if (aPredicatedElementList.size() == 2) {
            this.myHandleStatements[theHandleIndex] =
                theParser.getAddAt();
            theParser.setAddAt();
            theParser
                .getEstrelModule(theParser.getActualEstrelModule())
                .addStatementToModule(theParser.getAddAt(),
                    theParser.createStatement(((Element)
                        (aPredicatedElementList.get(0)))),
                    this.theHandleEvents[theHandleIndex] =
                        new EstrelSignalExpression(
                            "/*[@id=" +
                            + ((Element) aPredicatedElementList
                                .get(1)))
                            .getAttributeValue("id")
                            + "*//*",
                            theParser.getXMLDocument());
            if (this.theHandleEvents[theHandleIndex]
                .isSignalIdentifier()) {
                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 EstrelParserException(
                this.getStatementName() +
                " exceptionhandler not found");
        }
    }
}
570
if (this.myHandleStatements != null) {
    //anElementIndex indexes the first PredicatedStatement element
    List theHandlerList =
        DOMHelpers.getElements(theParser.getXMLDocument(),
            this.getXPatchSearchString()
            + "[Local-name(.)=" +
            + ((Element) anElementList.get(anElementIndex))
            .getTagName()
            + "]");
}
520
510
500
490
480
470

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

294
580           } //for
      } //if
    } //setsubstatements
  /**
   * Implements the abstract methode from <code>EsterelStatement</code>
   * and Returns a string representation of an esterel statement.
   * Sideeffects: none
   */
  @param anEsterelModule
  * @param anEsterelModule
  * @return an esterel modul representation
  * @see kiel.fileinterface.esterel.esterel2estudio.EsterelStatement
  */
  public final String toString(
    final EsterelModule anEsterelModule) {
    String aExceptionList = "";
    for (int i = 0; i < this.myExceptionSignals.length; i++) {
      aExceptionList += this.myExceptionSignals[i]
        .toString(anEsterelModule);
    }
    String result = this.getEsterelStatementName()
      + " "
      + aExceptionList
      + " in"
      + " "
      + ((EsterelStatement) anEsterelModule).getEsterelProgram()
        .get(this.theTrappedStatement).toString(anEsterelModule)
      + "\n";
  }
600
610           if (this.myHandleStatements != null) {
    for (int i = 0; i < this.myHandleStatements.length; i++) {
      result += "handle "
        + this.theHandleEvents[i].toString(anEsterelModule)
        + " do"
        + ((EsterelStatement) anEsterelModule)
          .getEsterelProgram()
            .get(this.myHandleStatements[i])
          + "\n";
    }
  }
620           }
  /**
   * @param anEsterelModule
   * @return a String representation of an esterel statement
   * @see kiel.fileinterface.esterel.esterel2estudio.EsterelStatement
   */
  public final String toString(
    final EsterelModule anEsterelModule) {
    String aExceptionList = "";
    for (int i = 0; i < this.myExceptionSignals.length; i++) {
      aExceptionList += this.myExceptionSignals[i]
        .toString(anEsterelModule);
    }
    String result = this.getEsterelStatementName()
      + " "
      + aExceptionList
      + " in"
      + " "
      + ((EsterelStatement) anEsterelModule).getEsterelProgram()
        .get(this.theTrappedStatement).toString(anEsterelModule)
      + "\n";
  }
630
640           /**
   * @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 verwendenden 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>
10   * <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 $
20   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;
    }
    /**
     * @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;
    }
    /**
     * checks if a optimization for <code>aState</code> is possible. Has
     * to be implemented in inherited classes.
     */
50
}

```

C.3.2. OptimizerChooser

Aufistung C.51: Die Klasse OptimizerChooser

```

private OptimizerChooser() {
    super();
}
/** 
 * abstract methode. starts the optimization of an statechart.
 */
30   * @param all
   *      if true then all levels will be optimized else only the
   *      lowest unoptimized level .
   */
40   * @param aChart
   *      a statechart
   * @return a statechart
   * @throws OptimizerException only delivers the exception.
   */
50   * @param aChart
   *      final StateChart aChart,
   *      final boolean all) throws OptimizerException {
   */
public static StateChart startOptimize(
    OptimizerProperties load()
{
    if (aChart != null) {
        Optimizer theOptimizer = null;
        if (aChart.getModeISource() .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>
 * $Revision: 1.25 $ last modified $Date: 2006/02/06 18:36:04 $
 */
public class OptimizerException extends Exception {
    /*
     * standard constructor.
     */
    public OptimizerException() {
        super();
    }
}

/*
 * OptimizerException
 * @param arg0 a string
 * @param arg1 a throwable
 */
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>
 * <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 $
 * </p>
 */
public abstract class Optimizer {
    /**
     * the statechart.
     */
    private StateChart aStateChart = null;

    /**
     * Standard constructor.
     */
    public Optimizer() {
        this(null);
    }

    /**
     * constructor which sets a statechart.
     */
    public Optimizer(StateChart aChart)
        throws OptimizerException;
}

/*
 * Optimizer interface
 */
public interface Optimizer {
    /**
     * Returns the aStateChart.
     */
    public final StateChart getStateChart()
        throws OptimizerException;
}

/*
 * Optimizer interface
 */
public interface Optimizer {
    /**
     * Returns the aStateChart.
     */
    public final StateChart getStateChart()
        throws OptimizerException;
}

/*
 * Optimizer interface
 */
public interface Optimizer {
    /**
     * Returns the aStateChart.
     */
    public final StateChart getStateChart()
        throws OptimizerException;
}

```

C.3.5. OptimizerProperties

Aufistung C.54: Die Klasse OptimizerProperties

```

        .equalsIgnoreCase("true");
    }

    /**
     * @return true, if option is set to true.
     */
    public static boolean getOptimizeESTEminateNeedlessNormalTransitions() {
        return properties.getProperty(
            OPTIMIZEESTELIMINATENEEDLESSNORMALTRANSITIONS
        ).equalsIgnoreCase("true");
    }

    /**
     * @return true, if option is set to true.
     */
    public static boolean getOptimizeESTEminateNeedlessSimpleStates() {
        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 getOptimizeESTJoinInFinalStates() {
        return properties.getProperty(
            OPTIMIZEESTJOININFINALSTATES
        ).equalsIgnoreCase("true");
    }

    /**
     * @return true, if option is set to true.
     */
    public static boolean getOptimizeESTSubs() {
        return properties.getProperty(OPTIMIZEESTSUBS);
    }

    /**
     * @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 optimizeESTEminateAbortedWithoutLocalsOrStates() {
        return properties.getProperty(
            OPTIMIZEESTELIMINATEABORTEDWITHOUTLOCALSSTATES
        ).equalsIgnoreCase("true");
    }

    /**
     * Copies properties to user specific file.
     */
    private static void copyDefaults() {
        InputStream is;
        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();
        } catch (IOException storeException) {
            success = false;
        }
        return success;
    }

    /**
     * Loads the defaults form resource file.
     */
    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.
     */
    private static void loadProperties() {
        boolean success = true;
        try {
            properties.load(new FileInputStream(
                PROPERTIESFILE
            ));
        } catch (IOException loadException) {
            success = false;
        }
        return success;
    }
}

```

C. Java Code für die Transformation von Esterel in SyncCharts

```
302
  /**
   * 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 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();
  }
}

220
230
240
```

C.4. Paket kiel.optimizer.estrelstudio

Das Paket `kiel.optimizer.estrelstudio` 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.

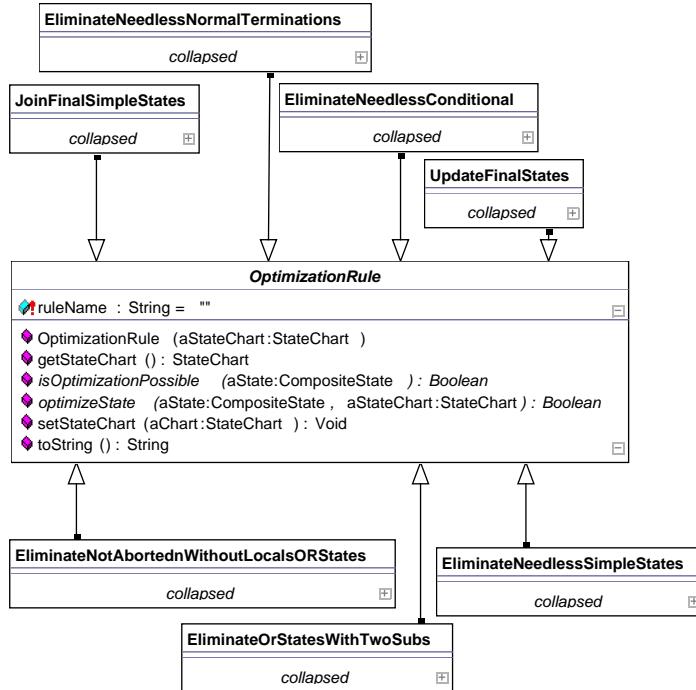


Abbildung C.37.: Klassendiagramm OptimizerRules

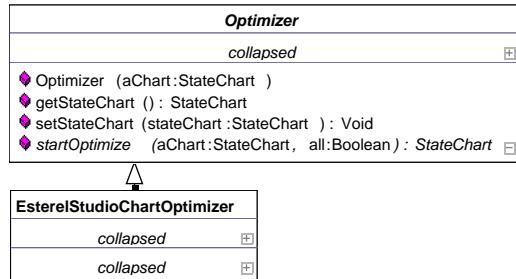


Abbildung C.38.: Klassendiagramm EsterelStudioChartOptimizer

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.Optimizer;
import kiel.optimizer.OptimizerException;
import kiel.util.CompoundLabelParser;
import kiel.util.StateChartHelpers;
import kiel.util.StateChartHelpers;

/**
 * Eliminates conditionals with only the default transition.
 */


* Copyright : Copyright (c) 2005
 * </p>
 * Company : Uni Kiel
 * </p>
 */
30  * @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");
    }
}

40 /**
 * checks if a optimization for <code>aState</code> is possible.
 */
50 * @param aState      a composite state
 * @return true if optimization is possible else false.
 */


```

C.4. Paket kiel.optimizer.estrelstudio

```

    .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 alTrans = (Transition) allIncomming
                    .get(j);
                TransitionLabel alabel = alTrans.getLabel();
                Action[] inActions = null;
                if (label 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) {
                abort = true;
            }
        }
    }
}

```

C. Java Code für die Transformation von Esterel in SyncCharts

1

2

C.4.2. EliminateNeedlessNormalTerminations

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.StateChart;
import kiel.dataStructure.Transition;
import kiel.optimizer.OptimizationRule;
import kiel.optimizer.OptimizerException;
import kiel.util.StatechartHelpers;

/*
 * <p>
 * Eliminates needless simple states.
 */
20   * <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 $
   */
30   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.
 */
40   public final boolean isOptimizationPossible(
        final CompositeState aState) {
    }

    /**
     * checks if a optimization for <code>aState</code> is possible.
     * @param aState
     *          a node
     * @return true if optimization is possible else false.
     */
50   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;
        }
    }

    /**
     * optimization for <code>aState</code>.
     */
80   public final void optimizeState(
        final CompositeState aState,
        final StateChart aStateChart)
    throws OptimizerException {
    }

    /**
     * @param result
     *          a composite state
     */
90   public final void optimizeState(
        final CompositeState aState,
        final StateChart aStateChart)
    throws OptimizerException {
    }

    /**
     * @param startAgain
     *          boolean startAgain = false;
     * @param aState
     *          a composite state
     */
100  public final void 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().getJsonValue());
                    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()
                                .getJsonValue()
                                + ((Transition) allOutTrans
                                    .get(0))
                                .getPriority()
                                .getJsonValue() - 1);
                        }
                    }
                }
            }
        }
    }
}
160
170
180
190
200
210
110
120
130
140
150
160
170
180
190
200
210
220
230
240
250
260
270
280
290
300
310
320
330
340
350
360
370
380
390
400
410
420
430
440
450
460
470
480
490
500
510
520
530
540
550
560
570
580
590
600
610
620
630
640
650
660
670
680
690
700
710
720
730
740
750
760
770
780
790
800
810
820
830
840
850
860
870
880
890
900
910
920
930
940
950
960
970
980
990
1000
1010
1020
1030
1040
1050
1060
1070
1080
1090
1100
1110
1120
1130
1140
1150
1160
1170
1180
1190
1200
1210
1220
1230
1240
1250
1260
1270
1280
1290
1300
1310
1320
1330
1340
1350
1360
1370
1380
1390
1400
1410
1420
1430
1440
1450
1460
1470
1480
1490
1500
1510
1520
1530
1540
1550
1560
1570
1580
1590
1600
1610
1620
1630
1640
1650
1660
1670
1680
1690
1700
1710
1720
1730
1740
1750
1760
1770
1780
1790
1800
1810
1820
1830
1840
1850
1860
1870
1880
1890
1900
1910
1920
1930
1940
1950
1960
1970
1980
1990
2000
2010
2020
2030
2040
2050
2060
2070
2080
2090
2100
2110
2120
2130
2140
2150
2160
2170
2180
2190
2200
2210
2220
2230
2240
2250
2260
2270
2280
2290
2300
2310
2320
2330
2340
2350
2360
2370
2380
2390
2400
2410
2420
2430
2440
2450
2460
2470
2480
2490
2500
2510
2520
2530
2540
2550
2560
2570
2580
2590
2600
2610
2620
2630
2640
2650
2660
2670
2680
2690
2700
2710
2720
2730
2740
2750
2760
2770
2780
2790
2800
2810
2820
2830
2840
2850
2860
2870
2880
2890
2900
2910
2920
2930
2940
2950
2960
2970
2980
2990
2995
3000
3005
3010
3015
3020
3025
3030
3035
3040
3045
3050
3055
3060
3065
3070
3075
3080
3085
3090
3095
3100
3105
3110
3115
3120
3125
3130
3135
3140
3145
3150
3155
3160
3165
3170
3175
3180
3185
3190
3195
3200
3205
3210
3215
3220
3225
3230
3235
3240
3245
3250
3255
3260
3265
3270
3275
3280
3285
3290
3295
3300
3305
3310
3315
3320
3325
3330
3335
3340
3345
3350
3355
3360
3365
3370
3375
3380
3385
3390
3395
3400
3405
3410
3415
3420
3425
3430
3435
3440
3445
3450
3455
3460
3465
3470
3475
3480
3485
3490
3495
3500
3505
3510
3515
3520
3525
3530
3535
3540
3545
3550
3555
3560
3565
3570
3575
3580
3585
3590
3595
3600
3605
3610
3615
3620
3625
3630
3635
3640
3645
3650
3655
3660
3665
3670
3675
3680
3685
3690
3695
3700
3705
3710
3715
3720
3725
3730
3735
3740
3745
3750
3755
3760
3765
3770
3775
3780
3785
3790
3795
3800
3805
3810
3815
3820
3825
3830
3835
3840
3845
3850
3855
3860
3865
3870
3875
3880
3885
3890
3895
3900
3905
3910
3915
3920
3925
3930
3935
3940
3945
3950
3955
3960
3965
3970
3975
3980
3985
3990
3995
4000
4005
4010
4015
4020
4025
4030
4035
4040
4045
4050
4055
4060
4065
4070
4075
4080
4085
4090
4095
4100
4105
4110
4115
4120
4125
4130
4135
4140
4145
4150
4155
4160
4165
4170
4175
4180
4185
4190
4195
4200
4205
4210
4215
4220
4225
4230
4235
4240
4245
4250
4255
4260
4265
4270
4275
4280
4285
4290
4295
4300
4305
4310
4315
4320
4325
4330
4335
4340
4345
4350
4355
4360
4365
4370
4375
4380
4385
4390
4395
4400
4405
4410
4415
4420
4425
4430
4435
4440
4445
4450
4455
4460
4465
4470
4475
4480
4485
4490
4495
4500
4505
4510
4515
4520
4525
4530
4535
4540
4545
4550
4555
4560
4565
4570
4575
4580
4585
4590
4595
4600
4605
4610
4615
4620
4625
4630
4635
4640
4645
4650
4655
4660
4665
4670
4675
4680
4685
4690
4695
4700
4705
4710
4715
4720
4725
4730
4735
4740
4745
4750
4755
4760
4765
4770
4775
4780
4785
4790
4795
4800
4805
4810
4815
4820
4825
4830
4835
4840
4845
4850
4855
4860
4865
4870
4875
4880
4885
4890
4895
4900
4905
4910
4915
4920
4925
4930
4935
4940
4945
4950
4955
4960
4965
4970
4975
4980
4985
4990
4995
5000
5005
5010
5015
5020
5025
5030
5035
5040
5045
5050
5055
5060
5065
5070
5075
5080
5085
5090
5095
5100
5105
5110
5115
5120
5125
5130
5135
5140
5145
5150
5155
5160
5165
5170
5175
5180
5185
5190
5195
5200
5205
5210
5215
5220
5225
5230
5235
5240
5245
5250
5255
5260
5265
5270
5275
5280
5285
5290
5295
5300
5305
5310
5315
5320
5325
5330
5335
5340
5345
5350
5355
5360
5365
5370
5375
5380
5385
5390
5395
5400
5405
5410
5415
5420
5425
5430
5435
5440
5445
5450
5455
5460
5465
5470
5475
5480
5485
5490
5495
5500
5505
5510
5515
5520
5525
5530
5535
5540
5545
5550
5555
5560
5565
5570
5575
5580
5585
5590
5595
5600
5605
5610
5615
5620
5625
5630
5635
5640
5645
5650
5655
5660
5665
5670
5675
5680
5685
5690
5695
5700
5705
5710
5715
5720
5725
5730
5735
5740
5745
5750
5755
5760
5765
5770
5775
5780
5785
5790
5795
5800
5805
5810
5815
5820
5825
5830
5835
5840
5845
5850
5855
5860
5865
5870
5875
5880
5885
5890
5895
5900
5905
5910
5915
5920
5925
5930
5935
5940
5945
5950
5955
5960
5965
5970
5975
5980
5985
5990
5995
6000
6005
6010
6015
6020
6025
6030
6035
6040
6045
6050
6055
6060
6065
6070
6075
6080
6085
6090
6095
6100
6105
6110
6115
6120
6125
6130
6135
6140
6145
6150
6155
6160
6165
6170
6175
6180
6185
6190
6195
6200
6205
6210
6215
6220
6225
6230
6235
6240
6245
6250
6255
6260
6265
6270
6275
6280
6285
6290
6295
6300
6305
6310
6315
6320
6325
6330
6335
6340
6345
6350
6355
6360
6365
6370
6375
6380
6385
6390
6395
6400
6405
6410
6415
6420
6425
6430
6435
6440
6445
6450
6455
6460
6465
6470
6475
6480
6485
6490
6495
6500
6505
6510
6515
6520
6525
6530
6535
6540
6545
6550
6555
6560
6565
6570
6575
6580
6585
6590
6595
6600
6605
6610
6615
6620
6625
6630
6635
6640
6645
6650
6655
6660
6665
6670
6675
6680
6685
6690
6695
6700
6705
6710
6715
6720
6725
6730
6735
6740
6745
6750
6755
6760
6765
6770
6775
6780
6785
6790
6795
6800
6805
6810
6815
6820
6825
6830
6835
6840
6845
6850
6855
6860
6865
6870
6875
6880
6885
6890
6895
6900
6905
6910
6915
6920
6925
6930
6935
6940
6945
6950
6955
6960
6965
6970
6975
6980
6985
6990
6995
7000
7005
7010
7015
7020
7025
7030
7035
7040
7045
7050
7055
7060
7065
7070
7075
7080
7085
7090
7095
7100
7105
7110
7115
7120
7125
7130
7135
7140
7145
7150
7155
7160
7165
7170
7175
7180
7185
7190
7195
7200
7205
7210
7215
7220
7225
7230
7235
7240
7245
7250
7255
7260
7265
7270
7275
7280
7285
7290
7295
7300
7305
7310
7315
7320
7325
7330
7335
7340
7345
7350
7355
7360
7365
7370
7375
7380
7385
7390
7395
7400
7405
7410
7415
7420
7425
7430
7435
7440
7445
7450
7455
7460
7465
7470
7475
7480
7485
7490
7495
7500
7505
7510
7515
7520
7525
7530
7535
7540
7545
7550
7555
7560
7565
7570
7575
7580
7585
7590
7595
7600
7605
7610
7615
7620
7625
7630
7635
7640
7645
7650
7655
7660
7665
7670
7675
7680
7685
7690
7695
7700
7705
7710
7715
7720
7725
7730
7735
7740
7745
7750
7755
7760
7765
7770
7775
7780
7785
7790
7795
7800
7805
7810
7815
7820
7825
7830
7835
7840
7845
7850
7855
7860
7865
7870
7875
7880
7885
7890
7895
7900
7905
7910
7915
7920
7925
7930
7935
7940
7945
7950
7955
7960
7965
7970
7975
7980
7985
7990
7995
8000
8005
8010
8015
8020
8025
8030
8035
8040
8045
8050
8055
8060
8065
8070
8075
8080
8085
8090
8095
8100
8105
8110
8115
8120
8125
8130
8135
8140
8145
8150
8155
8160
8165
8170
8175
8180
8185
8190
8195
8200
8205
8210
8215
8220
8225
8230
8235
8240
8245
8250
8255
8260
8265
8270
8275
8280
8285
8290
8295
8300
8305
8310
8315
8320
8325
8330
8335
8340
8345
8350
8355
8360
8365
8370
8375
8380
8385
8390
8395
8400
8405
8410
8415
8420
8425
8430
8435
8440
8445
8450
8455
8460
8465
8470
8475
8480
8485
8490
8495
8500
8505
8510
8515
8520
8525
8530
8535
8540
8545
8550
8555
8560
8565
8570
8575
8580
8585
8590
8595
8600
8605
8610
8615
8620
8625
8630
8635
8640
8645
8650
8655
8660
8665
8670
8675
8680
8685
8690
8695
8700
8705
8710
8715
8720
8725
8730
8735
8740
8745
8750
8755
8760
8765
8770
8775
8780
8785
8790
8795
8800
8805
8810
8815
8820
8825
8830
8835
8840
8845
8850
8855
8860
8865
8870
8875
8880
8885
8890
8895
8900
8905
8910
8915
8920
8925
8930
8935
8940
8945
8950
8955
8960
8965
8970
8975
8980
8985
8990
8995
9000
9005
9010
9015
9020
9025
9030
9035
9040
9045
9050
9055
9060
9065
9070
9075
9080
9085
9090
9095
9100
9105
9110
9115
9120
9125
9130
9135
9140
9145
9150
9155
9160
9165
9170
9175
9180
9185
9190
9195
9200
9205
9210
9215
9220
9225
9230
9235
9240
9245
9250
9255
9260
9265
9270
9275
9280
9285
9290
9295
9300
9305
9310
9315
9320
9325
9330
9335
9340
9345
9350
9355
9360
9365
9370
9375
9380
9385
9390
9395
9400
9405
9410
9415
9420
9425
9430
9435
9440
9445
9450
9455
9460
9465
9470
9475
9480
9485
9490
9495
9500
9505
9510
9515
9520
9525
9530
9535
9540
9545
9550
9555
9560
9565
9570
9575
9580
9585
9590
9595
9600
9605
9610
9615
9620
9625
9630
9635
9640
9645
9650
9655
9660
9665
9670
9675
9680
9685
9690
9695
9700
9705
9710
9715
9720
9725
9730
9735
9740
9745
9750
9755
9760
9765
9770
9775
9780
9785
9790
9795
9800
9805
9810
9815
9820
9825
9830
9835
9840
9845
9850
9855
9860
9865
9870
9875
9880
9885
9890
9895
9900
9905
9910
9915
9920
9925
9930
9935
9940
9945
9950
9955
9960
9965
9970
9975
9980
9985
9990
9995
9999
10000
10001
10002
10003
10004
10005
10006
10007
10008
10009
100010
100011
100012
100013
100014
100015
100016
100017
100018
100019
100020
100021
100022
100023
100024
100025
100026
100027
100028
100029
100030
100031
100032
100033
100034
100035
100036
100037
100038
100039
100040
100041
100042
100043
100044
100045
100046
100047
100048
100049
100050
100051
100052
100053
100054
100055
100056
100057
100058
100059
100060
100061
100062
100063
100064
100065
100066
100067
100068
100069
100070
100071
100072
100073
100074
100075
100076
100077
100078
100079
100080
100081
100082
100083
100084
100085
100086
100087
100088
100089
100090
100091
100092
100093
100094
100095
100096
100097
100098
100099
1000100
1000101
1000102
1000103
1000104
1000105
1000106
1000107
1000108
1000109
1000110
1000111
1000112
1000113
1000114
1000115
1000116
1000117
1000118
1000119
1000120
1000121
1000122
1000123
1000124
1000125
1000126
1000127
1000128
1000129
1000130
1000131
1000132
1000133
1000134
1000135
1000136
1000137
1000138
1000139
1000140
1000141
1000142
1000143
1000144
1000145
1000146
1000147
1000148
1000149
1000150
1000151
1000152
1000153
1000154
1000155
1000156
1000157
1000158
1000159
1000160
1000161
1000162
1000163
1000164
1000165
1000166
1000167
1000168
1000169
1000170
1000171
1000172
1000173
1000174
1000175
1000176
1000177
1000178
1000179
1000180
1000181
1000182
1000183
1000184
1000185
1000186
1000187
1000188
1000189
1000190
1000191
1000192
1000193
1000194
1000195
1000196
1000197
1000198
1000199
1000200
1000201
1000202
1000203
1000204
1000205
1000206
1000207
1000208
1000209
1000210
1000211
1000212
1000213
1000214
1000215
1000216
1000217
1000218
1000219
1000220
1000221
1000222
1000223
1000224
1000225
1000226
1000227
1000228
1000229
1000230
1000231
1000232
1000233
1000234
1000235
1000236
1000237
1000238
1000239
1000240
1000241
1000242
1000243
1000244
1000245
1000246
1000247
1000248
1000249
1000250
1000251
1000252
1000253
1000254
1000255
1000256
1000257
1000258
1000259
1000260
1000261
1000262
1000263
1000264
1000265
1000266
1000267
1000268
1000269
1000270
1000271
1000272
1000273
1000274
1000275
1000276
1000277
1000278
1000279
1000280
1000281
1000282
1000283
1000284
1000285
1000286
1000287
1000288
1000289
1000290
1000291
1000292
1000293
1000294
1000295
1000296
1000297
1000298
1000299
1000300
1000301
1000302
1000303
1000304
1000305
1000306
1000307
1
```

C. Java Code für die Transformation von Esterel in SyncCharts

```

        ((Transition) allInTrans.get(0))
            .getTarget()
            .removeIncomingTransition(
                ((Transition) allInTrans
                    .get(0)));
    } else {
        abort = true;
    }
    if (!abort) {
        useless.getTarget().removeIncomingTransition(
            useless);
        useless.getSource().removeOutgoingTransition(
            useless);
        startAgain = true;
        result = true;
    }
}
} // /substates
}
if (result) {
    StateChartHelpers.eliminateDeadStates(aState);
}
return result;
}

/**
 * checks if the two transition are fusionable.
 *
 * @param a
 *         incomingTransition
 * @param b
 *         outgoing Transition
 *
 * @param aState
 *         a composite state
 */
private boolean isFusionable(
    final Transition a,
    final CompositeState aState) {
    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
            if ((StateChartHelpers.compareTriggerWithoutCounter(
                a.getLabel(),
                b.getLabel(),
                this.getStatechart(),
                aState),
                StateChartHelpers.getTrigger(
                    b.getLabel(),
                    this.getStatechart(),
                    this.getStatechart(),
                    aState)))
                && StateChartHelpers.getEffect(
                    a.getLabel(),
                    aState).getActions().length == 0
                    && !StateChartHelpers.isDefaultTransition(a)
                    && ((a.getSource() instanceof State
                        && ((State) a.getSource()).getEntry() == null)
                        && (a.getSource().length == 0
                            && ((State) a.getSource()).getExit() == null)
                        && ((State) a.getSource().length == 0
                            && ((State) a.getSource()).getDoActivity() == null
                            && ((State) a.getSource().length == 0
                                && !((a.getSource() instanceof State))
                                && (b.getLabel() instanceof CompoundLabel
                                    && StateChartHelpers
                                        .withoutTrigger((CompoundLabel) b.getLabel())))
                                ) {
                    result = true;
                }
            return result;
        }
    }
}

```

C.4.4. EliminateNotAbortednWithoutLocalsORStates

Aufstellung C.58: Die Klasse EliminateNotAbortednWithoutLocalsORStates

```

package kiel.optimizer.estrelstudio;
import java.util.ArrayList;
import kiel.dataStructure.ANDState;
import kiel.dataStructure.CompositeState;
import kiel.dataStructure.OrState;
import kiel.dataStructure.StateChart;
import kiel.dataStructure.Transition;
import kiel.optimizer.OptimizationRule;
import kiel.optimizer.OptimizerException;
import kiel.util.StatechartHelpers;
import kiel.util.StatechartHelpers;

/**
 * <p>
 * Eliminates an ORState if it has no outgoing abortions 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
     * @param 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
     */
    public void doOptimizationForCompositeState(ANDState aState) {
        if (aState instanceof ORState) {
            if ((aState.getActions().length == 0
                && aState.getEntry().getActions().length == 0
                && aState.getExit().getActions().length == 0
                && aState.getLocalEvents().size() == 0
                && aState.getVariables().size() == 0
                && aState.getIncommingSuspension(aState).size() == 0
                && aState.getOutgoingAbortions(aState).size() == 0
                && aState.getSubStates().size() == 0) {
                if ((* (StatemachineHelpers.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
     */
    public void doOptimizationForCompositeState(StateChart aChart) {
        if (aChart instanceof ORState) {
            if ((aChart.getActions().length == 0
                && aChart.getEntry().getActions().length == 0
                && aChart.getExit().getActions().length == 0
                && aChart.getLocalEvents().size() == 0
                && aChart.getVariables().size() == 0
                && aChart.getIncommingSuspension(aChart).size() == 0
                && aChart.getOutgoingAbortions(aChart).size() == 0
                && aChart.getSubStates().size() == 0) {
                if ((* (StatemachineHelpers.getSubStates(aChart).size() <
                    * this.minNumberofSubStates) &&
                    * aChart instanceof ORState) {
                    return true;
                }
            }
            return false;
        }
    }
}

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

/*
public final boolean optimizeState(
    final CompositeState aState,
    final StateChart aStatechart)
throws OptimizerException {
    boolean result = false;
    this.setStateChart(aStatechart);
    if (aState instanceof ORState) {
        result = true;
        StateChartHelpers
            .deleteOrStateSubStates((ORState) aState);
    } else if (aState instanceof ANDState) {
        if (aState getParent() != null) {
            ArrayList<Object> in =
                (ArrayList<Object>) aState getParent()
                    .getIncomingTransitions();
            ArrayList<Object> out =
                (ArrayList<Object>) 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()
                    .removeIncomingTransition(
                        ((Transition) out.get(0)));
            }
        }
    }
    return result;
}

```

C.4.5. EliminateOrStatesWithTwoSubs

Auflistung C.59: Die Klasse EliminateOrStatesWithTwoSubs

```

package kiel.optimizer.estereelstudio;

import java.util.ArrayList;
import kiel.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.StateAbort;
import kiel.optimizer.OptimizationRule;
import kiel.optimizer.OptimizerException;
import kiel.util.StatechartHelpers;
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("EliminateOrStatesWithTwoSubs");
    }
}

/*
 * checks if a optimization for <code>aState</code> is possible.
 */
* @param aState a composite state
*/
100   /* Eliminates macros with only two substates.
 * @param aState the macrostate.
 * @param aStateChart the StateChart.
*/

```

C. Java Code für die Transformation von Esterel in SyncCharts

```

* @return true, if something was optimized.
* @throws OptimizerException if something
* unexpected happened.
*/
public boolean optimizeState(
    final CompositeState aState,
    final StateChart aStatechart)
throws OptimizerException {
    this.setStatechart(aStatechart);
    if (isOptimizationPossible(aState)) {
        if (aState.getParent() != null) {
            StateChartHelpers.deleteOrSaveSubStates((ORState) aState);
        }
        return true;
    }
    State aSubState = StateChartHelpers.getSubStates(aState).get(0);
    110
    120
    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());
        aSubState.setRootode((CompositeState) aSubState);
        aSubState.setParent(null);
        aSubState.setName(aState.getName());
        return true;
    }
    130
    } //if
    return false;
} //method
}

```

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>
 * <p>
 * @author <a href="mailto:lku@informatik.uni-kiel.de">Lars Kueh</a>
 */
public final class EsterelStudioChartOptimizer
    extends Optimizer {
    /**
     * the actual optimized level.
     */
    private int actualLevel = -1;

    /**
     * a flat representation of an statechart sorted by the depth of a
     * state.
     */
    private StateChartDataAttributes[] myChart = null;

    /**
     * An array with rules sorted by invocation in a state.
     */
    private OptimizationRule[] theRules = null;

    /**
     * the standard constructor.
     */
    /**
     * @throws OptimizerException
     *          delivers it.
     */
    public EsterelStudioChartOptimizer()
        throws OptimizerException {
    }

    /**
     * a constructor which sets the statechart class variable.
     */
    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.
     */
    public void all
        (@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.
        )
        (@Param aChart
         a statechart
         final boolean all)
        throws OptimizerException {
        boolean aChange = false;
        this.actualLevel = -1;
        this.myChart = null;
        String lastUsedRule = "";
        if (aChart != null) {
            if (this.myChart == null) {
                this.setMyChart(aChart);
            }
            long memory=0;
            boolean round = true;
            int deep = -1;
        }
    }

    /**
     * An array with rules sorted by invocation in a state.
     */
    /**
     * @throws OptimizerException
     *          delivers it.
     */
    public void startOptimize(
        final Statechart aChart,
        final boolean all
    )
        throws OptimizerException {
        boolean aChange = false;
        this.actualLevel = -1;
        this.myChart = null;
        String lastUsedRule = "";
        if (aChart != null) {
            if (this.myChart == null) {
                this.setMyChart(aChart);
            }
            long memory=0;
            boolean round = true;
            int deep = -1;
        }
    }

    /**
     * An array with rules sorted by invocation in a state.
     */
    /**
     * @throws OptimizerException
     *          delivers it.
     */
    public void startOptimize(
        final Statechart aChart,
        final boolean all
    )
        throws OptimizerException {
        boolean aChange = false;
        this.actualLevel = -1;
        this.myChart = null;
        String lastUsedRule = "";
        if (aChart != null) {
            if (this.myChart == null) {
                this.setMyChart(aChart);
            }
            long memory=0;
            boolean round = true;
            int deep = -1;
        }
    }
}

```

C. Java Code für die Transformation von Esterel in SyncCharts

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) 2006
 * * </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 aChart
     */
    public JoinFinalSimpleStates(final StateChart 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;
        else
            return true;
    }

    /**
     * optimization for <code>aState</code>.
     * @param aState a composite state
     * @param aState a statechart
     * @return null
     * @throws OptimizerException
     *          not throw, may be thrown in called methods.
     * @see kiel.dataStructure.CompositeState
     */
    public final boolean optimizeState(FinalCompositeState 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(1)));
    }
}

```

C.4.8. StateChartNodeAttributes

Auflistung C.62: Die Klasse StateChartNodeAttributes

```

package kiel.optimizer.estereistudio;

import java.util.ArrayList;
import kiel.dataStructure.CompositeState;

/**
 * <p>
 * A class for get an flat structure from a statechart
 * where the Composite states know their depth.
 */
10   *
 * </p>
 * <p>
 * Copyright : Copyright (c) 2005
 * Company : Uni Kiel
 * </p>
 * <p>
 * @author <a href="mailto:liku@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 usedOptimizations = null;
}

/**
 * A constructor which sets the class variables
 * theCState and deep.
 */
40   *
 * @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.usedOptimizations = new ArrayList();
}

/**
 * @return Returns the deep.
 */
50   public final int getDeep() {
    return this.deep;
}

/**
 * @param adeep The deep to set.
 */
60   public final void setDeep(final int adeep) {
    this.deep = adeep;
}

/**
 * @return Returns the theCState.
 */
70   public final CompositeState getTheCState() {
    return this.theCState;
}

/**
 * @param acState The theCState to set.
 */
80   public final void setTheCState(final CompositeState acState) {
    this.theCState = acState;
}

/**
 * @return Returns the usedOptimizations .
 */
90   public final ArrayList getUsedOptimizations() {
    return this.usedOptimizations;
}

/**
 * @param unusedOptimizations The usedOptimizations to set .
 */
95   public final void setUsedOptimizations(
    final ArrayList unusedOptimizations) {
    this.usedOptimizations = unusedOptimizations;
}

/**
 * adds an rule name to usedOptimizations .
 */
100  public final void addOptimizationRule(
    final String ruleName) {
    this.usedOptimizations.add(ruleName);
}

/**
 * compares the object with arg0.
 */
100  *
 * @param arg0 a object.
 * @return 0 if equal , -1 if less else +1.
 */
100  public final int compareTo(final Object arg0) {
    int result = 0;
}

```


C.4.9. UpdateFinalStates

Auflistung C.63: Die Klasse UpdateFinalStates

```

package kiel.optimizer.estereistudio;

import java.util.ArrayList;
import kiel.dataStructure.ANDState;
import kiel.dataStructure.CompositeState;
import kiel.dataStructure.FinalANDState;
import kiel.dataStructure.FinalORSState;
import kiel.dataStructure.StateChart;
import kiel.optimizer.OptimizationRule;
import kiel.optimizer.OptimizerException;
import kiel.util.StateChartHelpers;

/*
 * Converts final states
 * which have no final substates to non final states.
 */
20  * Copyright: Copyright (c) 2005
 *             Lars Kuehl <lku@informatik.uni-kiel.de>
 *             Uni Kiel
 *             $Revision: 1.4 $ last modified $Date: 2006/02/17 20:00:22 $
 */
public class UpdateFinalStates
    extends OptimizationRule {
    /*
     * @param aChart
     * @param aState
     */
30    public void updateFinalStates(StateChart aChart,
        StateChart aState) {
        super(aChart);
        super.setRuleName("UpdateFinalStates");
    }
}

/*
 * checks if a optimization for aState is possible.
 */
40  * @param aState
 * @param aChart
 * @param aState
 */
45  public boolean isOptimizationPossible(
        final CompositeState aState) {
50      if (aState instanceof FinalANDState && aState.getParent() != null) {
55          if (StateChartHelpers.getSubFinalStates(aState).size() == 0) {
56              return true;
57          }
58      } else if (aState instanceof FinalORSState && aState.getParent() != null) {
59          ArrayList allRegions = ((ANDState) aState).getSubNodes();
60          for (int i = 0; i < allRegions.size(); i++) {
61              if (StateChartHelpers
62                  .getSubFinalStates(
63                      (CompositeState) allRegions
64                          .get(i))
65                          .size() == 0) {
66                  return true;
67              }
68          }
69      } else {
70          /*
71             * optimization for <code>aState</code>.
72             * @param aState
73             * @param aState
74             * @param aStateChart
75             * @param aStateChart
76             * @throws OptimizerException
77             * @see kiel.dataStructure.CompositeState
78             */
79          public final boolean optimizeState(
80              final CompositeState aState,
81              final StateChart aStateChart,
82              throws OptimizerException {
83              boolean result = false;
84              this.setStateChart(aStateChart);
85              if (isOptimizationPossible(aState)) {
86                  // If there is more than one final simple state.
87                  StateChartHelpers.convertFromFinalState(aState);
88              }
89          }
90      }
91      return result;
92  }
}

```