Efficient development of Statechart models
A comparative study

Reinhard von Hanxleden

Joint work with Steffen Prochnow, Jürgen Golz and KIEL contributors

Real-Time Systems and Embedded Systems Group
Department of Computer Science
Christian-Albrechts-Universität zu Kiel, Germany
www.informatik.uni-kiel.de/rtsys

SYNCHRON 2007
29 November 2007, Bamberg

Institut für Informatik
Christian-Albrechts-Universität zu Kiel

Overview

Graphical Modeling—A Good Thing!

> *Today, we see with some surprise that visual notations for synchronous languages have found their way to successful industrial use with the support of commercial vendors. This probably reveals that building a visual formalism on the top of a mathematically sound model gives actual strength to these formalisms and makes them attractive to users.*
>
> Benveniste et. al.

Albert Benveniste, Paul Caspi, Stephen A. Edwards, Nicolas Halbwachs, Paul Le Guernic, and Robert de Simone.
The Synchronous Languages Twelve Years Later.
In *Proceedings of the IEEE, Special Issue on Embedded Systems*, volume 91, pages 64–83, January 2003.

Graphical Modeling—A Good *Enough* Thing?

Graphical Modeling—A Good *Enough* Thing?

Observation 1:

- Graphical languages are convenient to browse

Graphical Modeling—A Good *Enough* Thing?

Observation 1:

- Graphical languages are convenient to browse
- . . . but are can be a pain to *edit*!

Graphical Modeling—A Good *Enough* Thing?

Observation 1:

- Graphical languages are convenient to browse
- . . . but are can be a pain to *edit*!

Observation 2:

- Graphical languages are appealing

Graphical Modeling—A Good *Enough* Thing?

Observation 1:

- Graphical languages are convenient to browse
- . . . but are can be a pain to *edit*!

Observation 2:

- Graphical languages are appealing
- . . . but not necessarily *effective* in conveying technical information!

Graphical Modeling—A Good *Enough* Thing?

Observation 1:

- Graphical languages are convenient to browse
- . . . but are can be a pain to *edit*!

Observation 2:

- Graphical languages are appealing
- . . . but not necessarily *effective* in conveying technical information!

Observation 3:

- Graphical models already work well to visualize complex structures

Graphical Modeling—A Good *Enough* Thing?

Observation 1:

- Graphical languages are convenient to browse
- . . . but are can be a pain to *edit*!

Observation 2:

- Graphical languages are appealing
- . . . but not necessarily *effective* in conveying technical information!

Observation 3:

- Graphical models already work well to visualize complex structures
- . . . but are still limited for visualizing complex *behaviors*!

Our Approach

The vision:

- Provide flexible, alternative views of system under development (SUD)
- Free the designer from tedious model editing tasks
- Combine best of graphical and textual worlds

Our Approach

The vision:

- Provide flexible, alternative views of system under development (SUD)
- Free the designer from tedious model editing tasks
- Combine best of graphical and textual worlds

The key enabler:

- Automatic, flexible synthesis of graphical models

Our Approach

The vision:

- Provide flexible, alternative views of system under development (SUD)
- Free the designer from tedious model editing tasks
- Combine best of graphical and textual worlds

The key enabler:

- Automatic, flexible synthesis of graphical models

The challenge:

- Automatic layout with “appealing” results

Our Approach

The vision:

- Provide flexible, alternative views of system under development (SUD)
- Free the designer from tedious model editing tasks
- Combine best of graphical and textual worlds

The key enabler:

- Automatic, flexible synthesis of graphical models

The challenge:

- Automatic layout with "appealing" results

Our experimental platform:

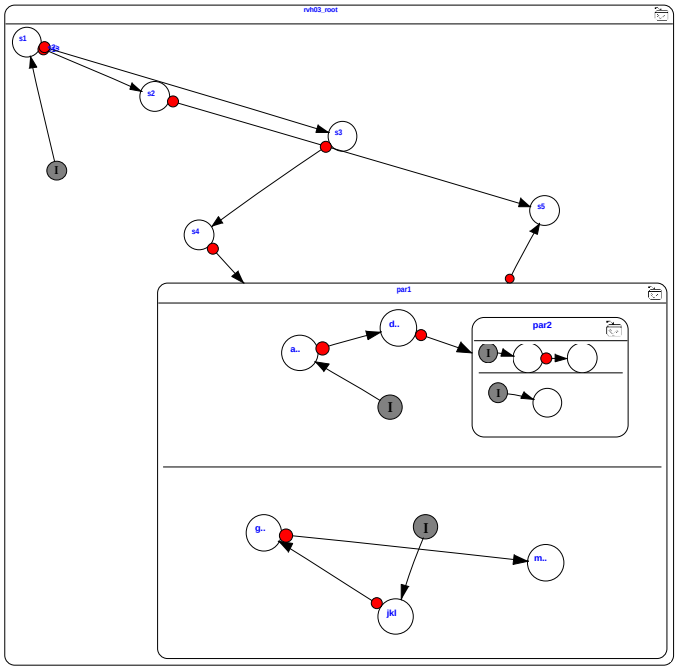
- Kiel Integrated Environment for Layout

Editing Statecharts in KIEL—2003

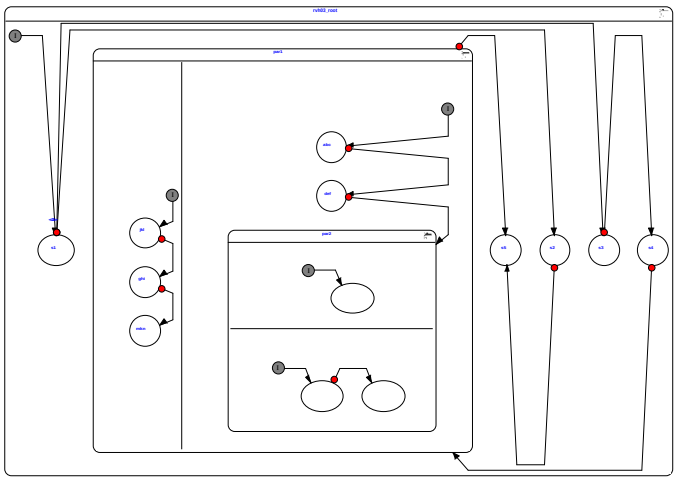
- Very simple horizontal/vertical layouter
- Import SyncCharts from Esterel Studio

Editing Statecharts in KIEL—2003

- Very simple horizontal/vertical layouter
- Import SyncCharts from Esterel Studio



(a) Original Layout



(b) After Auto-layout

KIEL—Current State

Automatic Layout:

- Employ various strategies (GraphViz + others)
- Use generic hierarchy wrapper

Model creation:

- Import from Esterel Studio, ArgoUML, Stateflow
- KIEL macro editor
- KIT Editor
- Synthesis from Esterel *LCTES'06*
- Robustness checking *MARTES'06*
- Cognitive experiments *MODELS'07*

Simulation:

- Use dynamic statecharts *DATE'06*

Overview

Creating Graphical Models

State of the Practice

- WYSIWYG editors to create graphical models
- Some editors offer alignment tools (ArgoUML)
- In initial phase, often resort to paper and pencil
- Creating and maintaining graphical models is time consuming

Creating Graphical Models

State of the Practice

- WYSIWYG editors to create graphical models
- Some editors offer alignment tools (ArgoUML)
- In initial phase, often resort to paper and pencil
- Creating and maintaining graphical models is time consuming

The Problem

- Non-linearity — Text: 1-D, Graphics: 2-D
- Context entanglement — Transitions, State hierarchy, concurrency

Alternatives to Accelerate Editing

1. Add-on to traditional editors
   - Create quick-and-dirty graphical model (WYSIWYG) ...
   - ... then apply automated layout

Alternatives to Accelerate Editing

1. Add-on to traditional editors
   - Create quick-and-dirty graphical model (WYSIWYG) ...
   - ... then apply automated layout
2. Macro-based modeling
   - Employs Statechart production rules
   - Incremental synthesis
   - Also referred to as "structure-based"

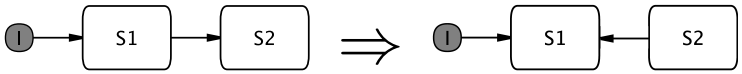
Alternatives to Accelerate Editing

1. Add-on to traditional editors
   - Create quick-and-dirty graphical model (WYSIWYG) ...
   - ... then apply automated layout
2. Macro-based modeling
   - Employs Statechart production rules
   - Incremental synthesis
   - Also referred to as "structure-based"
3. Text-based modeling
   - Modeler uses textual language
   - Model synthesis from textual description

Macro-Based Modeling

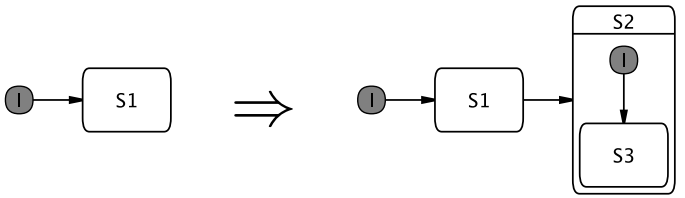
- Identified nine main Statechart editing schemata
- Three categories: 1. *creation*, 2. *modification*, 3. *deletion*

(a) Insertion of a simple successor state.

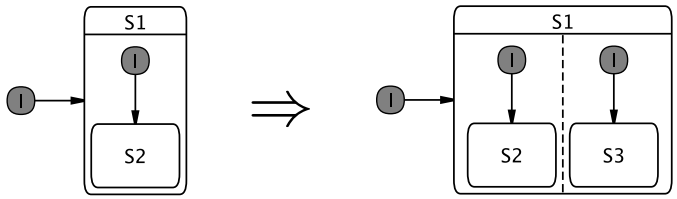


(b) Modification of transition direction.

(c) Deletion of a Statechart element.

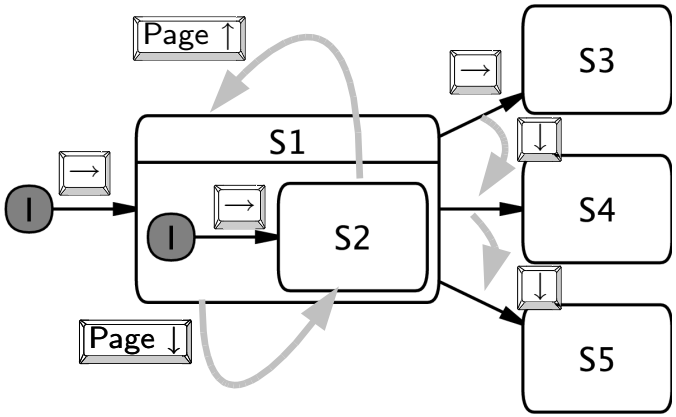


(d) Insertion of hierarchical successor state.

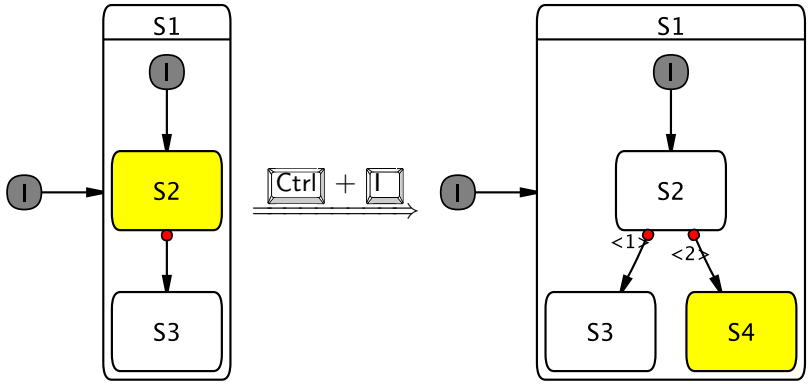


(e) Insertion of a parallel region.

Macro-Based Modeling



(a) Navigation with key strokes.



(b) Example of applying the "insert simple successor state" schema.

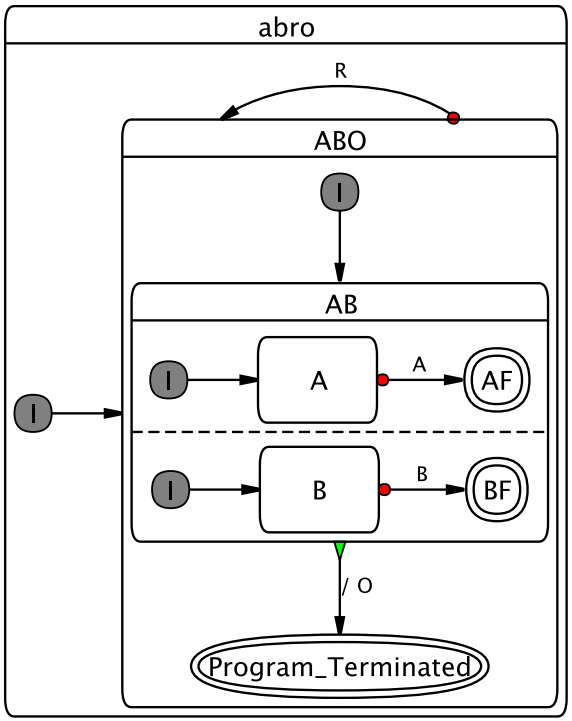
Text-Based Modeling

KIel Statechart extension of doT:

- Implicit declarations in *dot*,
- Hierarchy construction in *Argos*,
- Orthogonal construction in *Esterel*, and
- Ability to describe different Statechart dialects

```
statechart abro[model="Esterel Studio";version="5.0"]{
 input A;
 input B;
 input R;
 output O;
 {
  ->ABO;
  ABO{
   AB{
     ->A;
     A->AF[type=sa;label="A"];
     AF[type=final];
    ||
     ->B;
     B->BF[type=sa;label="B"];
     BF[type=final];
   };
   ->AB;
   AB->Program_Terminated[type=nt;label="/ O"];
   Program_Terminated[type=final];
  };
  ABO->ABO[type=sa;label="R"];
 };
};
```

(a) KIT description representation.



(b) SSM representation.

Visualizing Complex Behaviors

- View of single chart rarely suffices—need to see several active charts at once
- Set of active charts changes dynamically
- Keeping active charts in foreground requires significant additional user effort during simulation

The Problem:

- Concurrency
- Fixed level of detail
- Spreading system across several charts (windows) aids model creation and maintenance
  . . . but results in fragmented overall picture

Dynamic Charts

- Introduce different system views, defining
  - visible parts of the system
  - visible level of detail
- Present dynamically changing views dependent on
  1. Simulation state
  2. User requests
- A dynamic extension to semantic focus-and-context representation

Oliver Köth.
Semantisches Zoomen in Diagrammeditoren am Beispiel von UML.
Master's thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, 2001.

Overview

The Experiment

Goals

1. Investigation of differences in editing using an WYSIWYG Statechart editor, the KIEL macro editor, and the KIT editor
2. Comparison of the readability of Statechart layouts created by the KIEL layouter and other Statechart layouts

The Experiment

Goals

1. Investigation of differences in editing using an WYSIWYG Statechart editor, the KIEL macro editor, and the KIT editor
2. Comparison of the readability of Statechart layouts created by the KIEL layouter and other Statechart layouts

Subjects

- Graduate-level students attending the lecture "Model-Based Design and Distributed Real-Time Systems" in the Winter Semester 2006/07
- Conducted two series of experiments, at beginning and end of semester
  1. Novices (24 subjects): basic knowledge concerning Statecharts
  2. Advanced (19 subjects): some practical experience

Experiment 1: Statechart Creation

Hypotheses

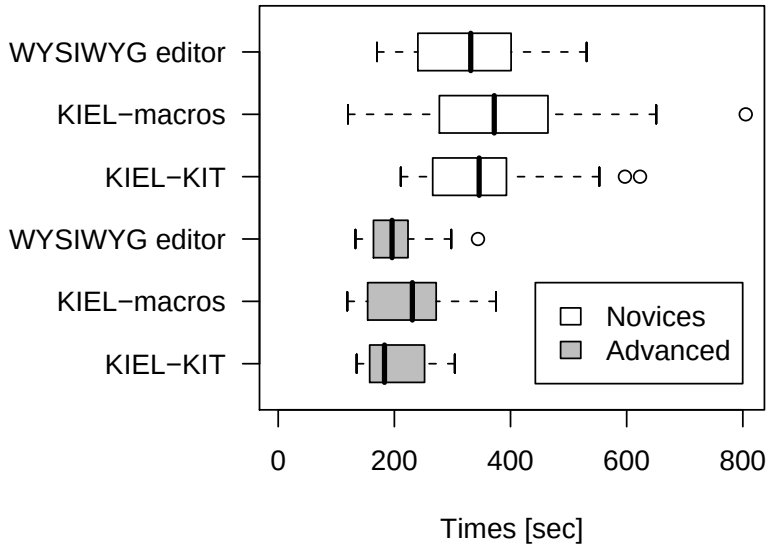
1. Novices will need less time to create a Statechart using the WYSIWYG editor.
2. Advanced will need less time using the KIT editor.

Experiment 1: Statechart Creation

Hypotheses

1. Novices will need less time to create a Statechart using the WYSIWYG editor.
2. Advanced will need less time using the KIT editor.

Results



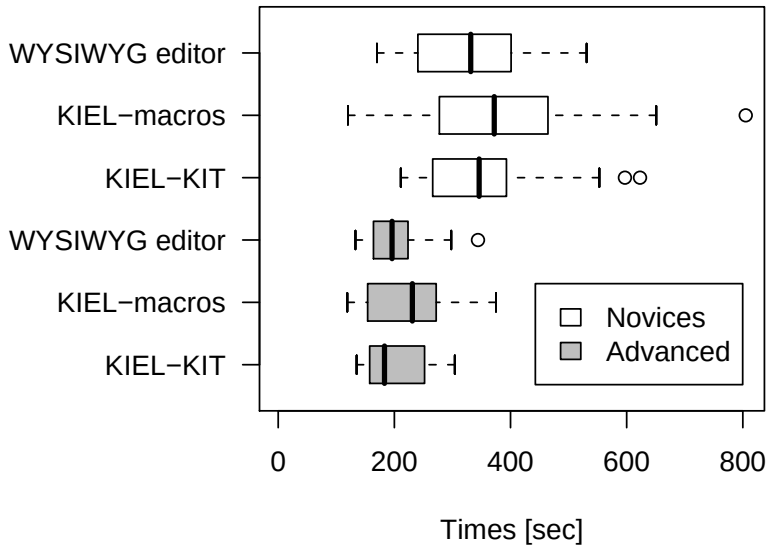
Distribution of times for creating a new Statechart

Experiment 1: Statechart Creation

Hypotheses

1. Novices will need less time to create a Statechart using the WYSIWYG editor.
   Not quite!
2. Advanced will need less time using the KIT editor.

Results



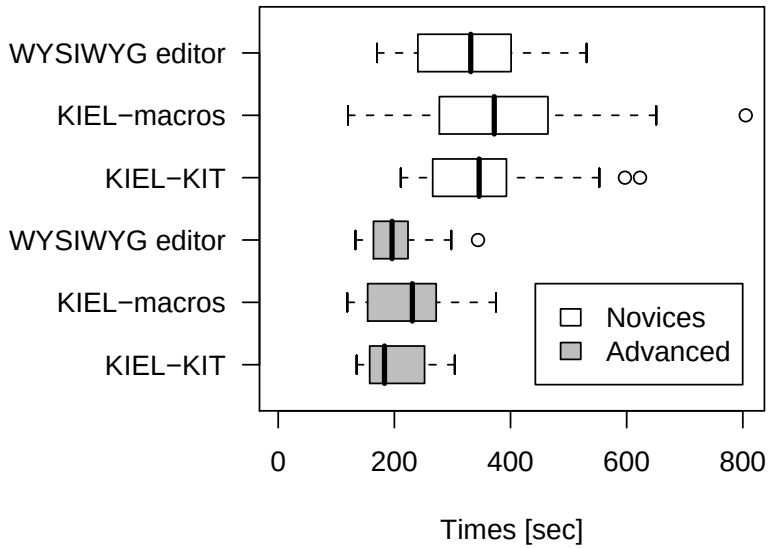
Distribution of times for creating a new Statechart

Experiment 1: Statechart Creation

Hypotheses

1. Novices will need less time to create a Statechart using the WYSIWYG editor.
   Not quite!
2. Advanced will need less time using the KIT editor.
   Weakly confirmed.

Results



Distribution of times for creating a new Statechart

Experiment 2: Statechart Modification

Hypothesis

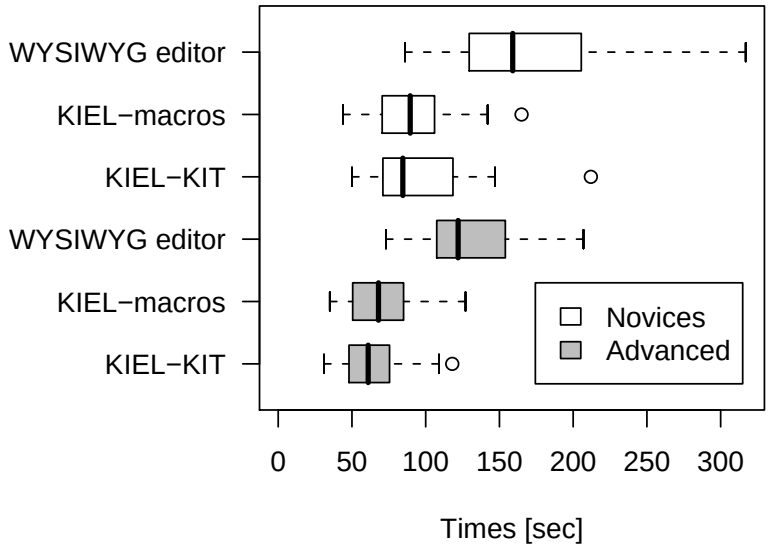
Statechart modification using the KIT editor or the KIEL macro editor is faster than using the WYSIWYG editor.

Experiment 2: Statechart Modification

Hypothesis

Statechart modification using the KIT editor or the KIEL macro editor is faster than using the WYSIWYG editor.

Results



Distribution of times for modifying an existing Statechart

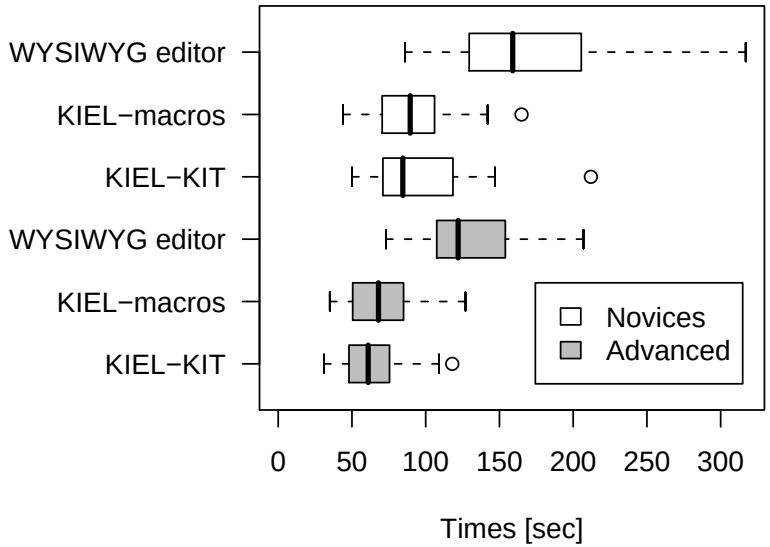
Experiment 2: Statechart Modification

Hypothesis

Statechart modification using the KIT editor or the KIEL macro editor is faster than using the WYSIWYG editor.

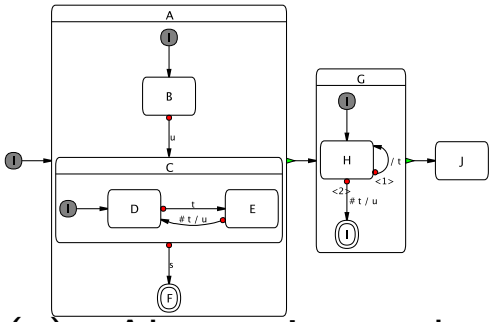
Confirmed.

Results

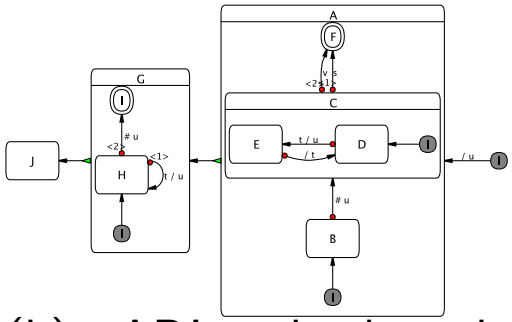


Distribution of times for modifying an existing Statechart

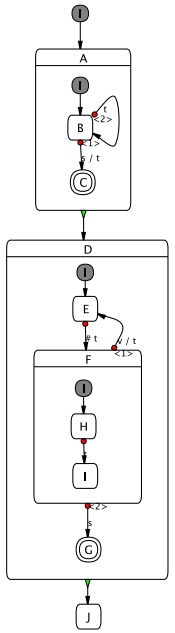
Statechart layout alternatives



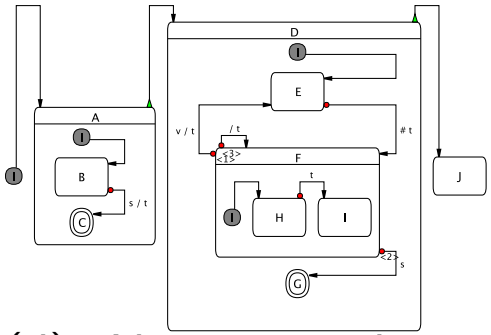
(a) Alternating dot layout (ADL)



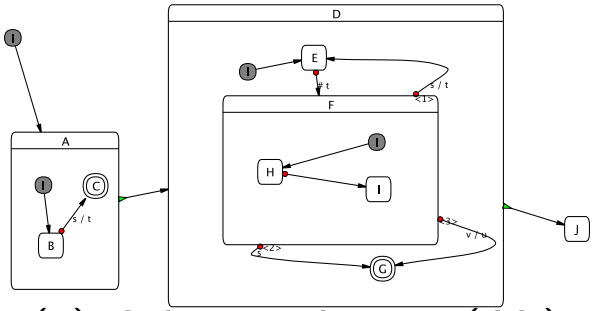
(b) ADL backwards (ADBL)



(c) Linear layer layout (LLL)



(d) Alternating linear layout (ALL)



(e) Arbitrary layout (AL)

Experiment 3: Statechart Aesthetics

Hypothesis

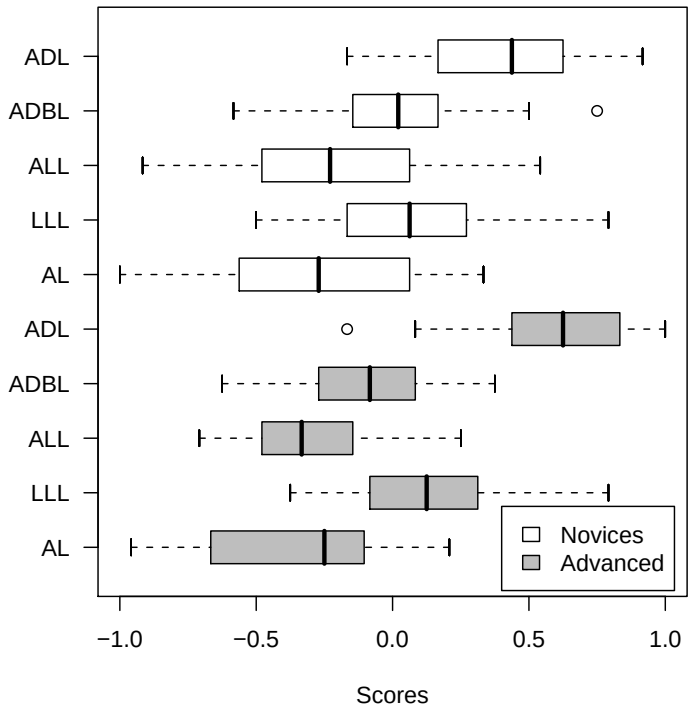
We expect the best scores for Statecharts laid out according certain layout styles realized by the KIEL Statechart layouter.

Experiment 3: Statechart Aesthetics

Hypothesis

We expect the best scores for Statecharts laid out according certain layout styles realized by the KIEL Statechart layouter.

Results



Distribution of subjective Statechart layout scores

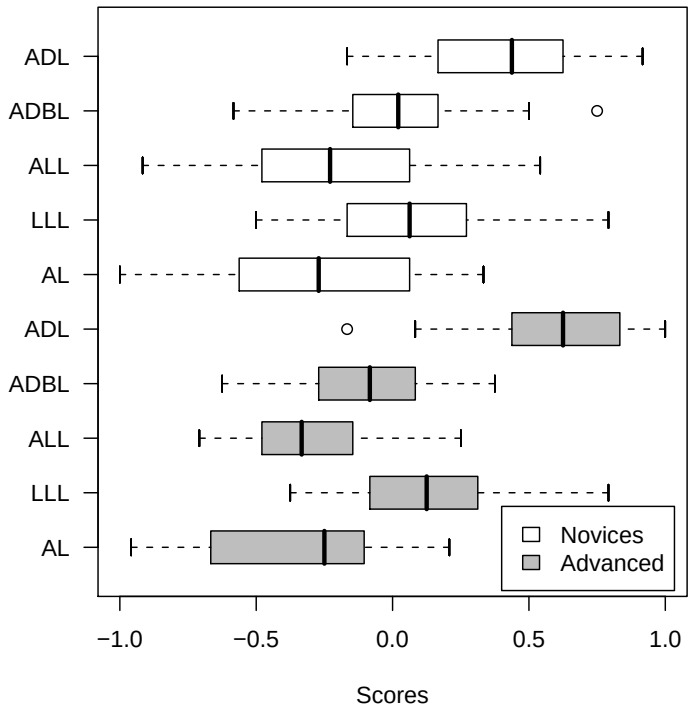
Experiment 3: Statechart Aesthetics

Hypothesis

We expect the best scores for Statecharts laid out according certain layout styles realized by the KIEL Statechart layouter.

Confirmed—and slightly more pronounced for advanced users.

Results



Distribution of subjective Statechart layout scores

Experiment 4: Statechart Comprehension

Hypothesis

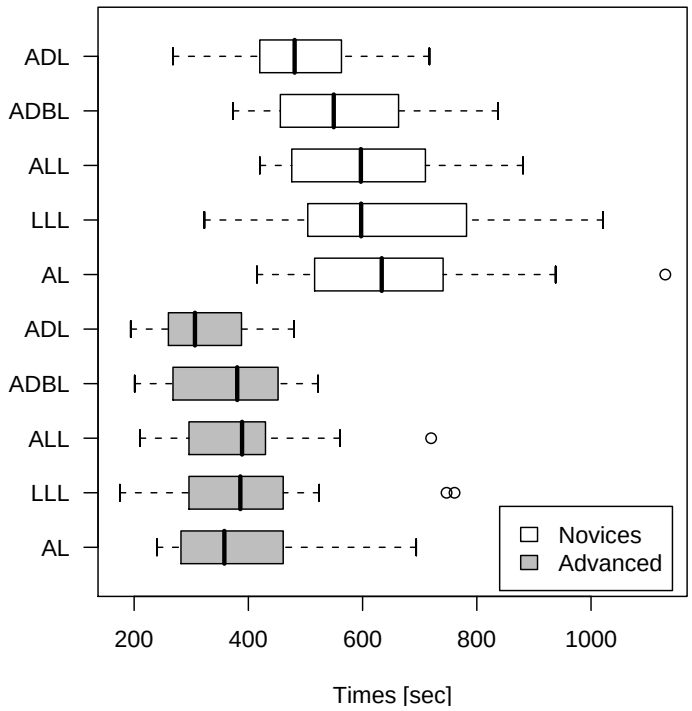
Well arranged Statecharts, as laid out by the KIEL Statechart layouter are better (faster) understandable than arbitrary layouts.

Experiment 4: Statechart Comprehension

Hypothesis

Well arranged Statecharts, as laid out by the KIEL Statechart layouter are better (faster) understandable than arbitrary layouts.

Results



Distribution of Statechart comprehension times

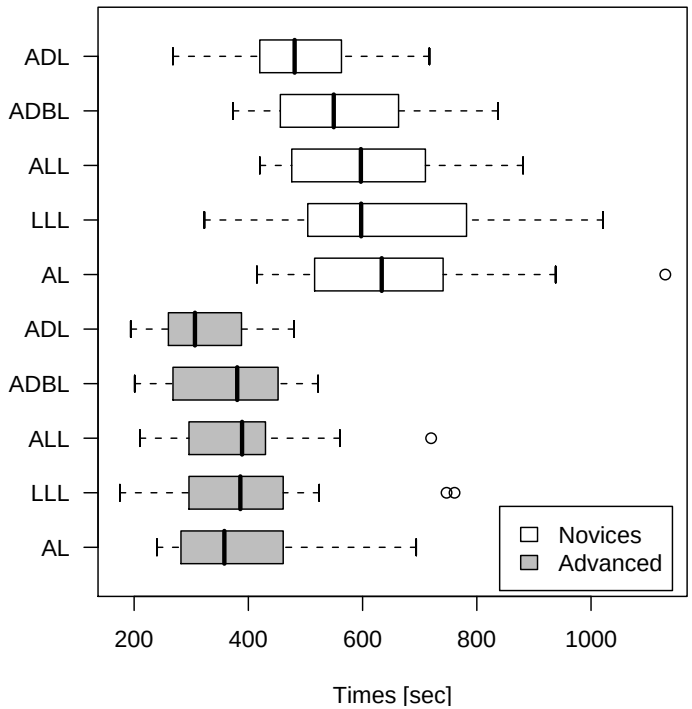
Experiment 4: Statechart Comprehension

Hypothesis

Well arranged Statecharts, as laid out by the KIEL Statechart layouter are better (faster) understandable than arbitrary layouts.

Confirmed—and more pronounced for novices.

Results



Distribution of Statechart comprehension times

Summary

- KIEL provides a platform for experimenting with the pragmatics of graphical modeling
- Have conducted experiments within the classroom
- Most, but not all, hypotheses were confirmed
- Overall results indicate usefulness of editing alternatives to the classical WYSIWYG paradigm
- Still missing: study with more complex, real-world designs

Outlook

Are transitioning to new generation of KIEL

- Make visual information truly first class citizen
- Incorporate data flow
- Make it as open as possible
- Many more ideas . . .

Open questions

- How to layout data flow diagrams?
- What about "dynamic" data flow?
- What platform to use? (Eclipse? Ptolemy? . . .)

Outlook

Are transitioning to new generation of KIEL

- Make visual information truly first class citizen
- Incorporate data flow
- Make it as open as possible
- Many more ideas . . .

Open questions

- How to layout data flow diagrams?
- What about "dynamic" data flow?
- What platform to use? (Eclipse? Ptolemy? . . .)

`www.informatik.uni-kiel.de/en/rtsys/kiel/`

Thanks! Questions or comments?