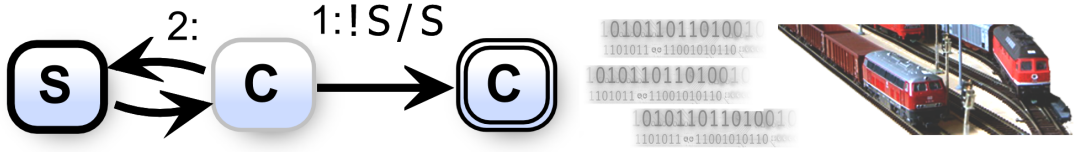
SCCharts: The Railway Project Report



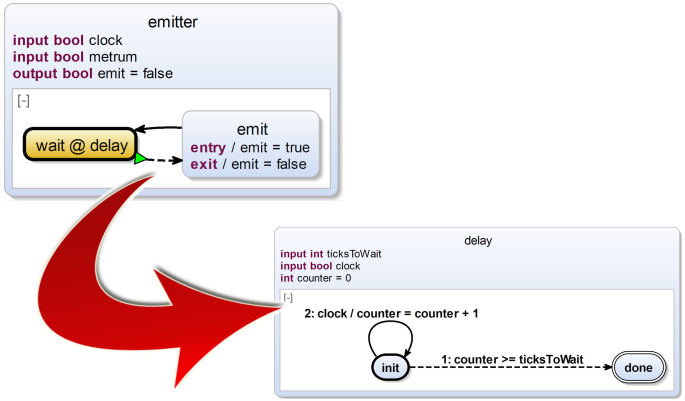
Steven Smyth

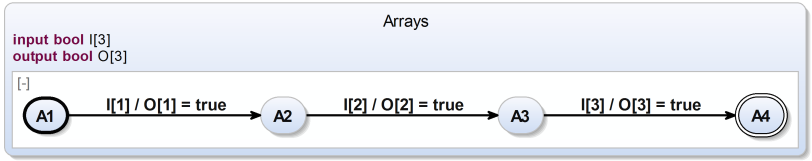
Real-Time Systems and Embedded Systems Group
Department of Computer Science
Kiel University, Germany

Oberseminar, SoSem 14
27.08.2014

Overview

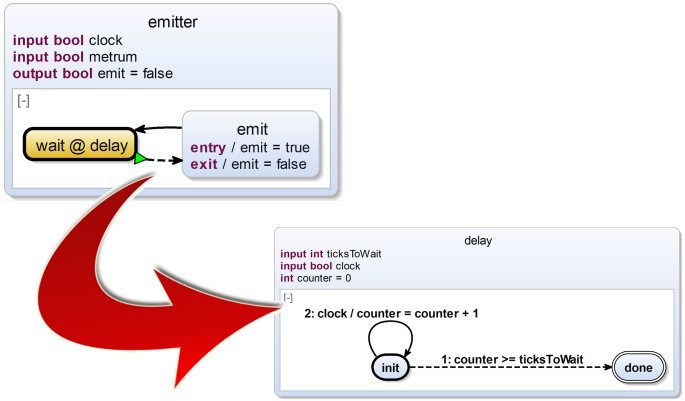
SCCharts Extensions - Roadmap

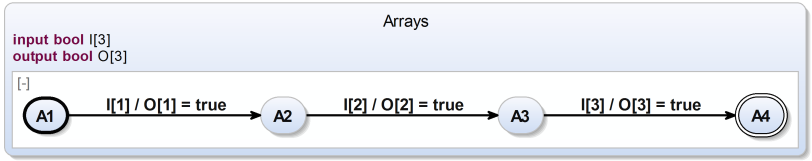




Category	Feature	OSem	Rail	Status
Core	Referenced SCCharts	✓		
	Map & For	✓		
Declaration	Arrays	✓		
	Const	✓		
	Extern			
	Structs	~		
Hostcode	Function calls	✓		

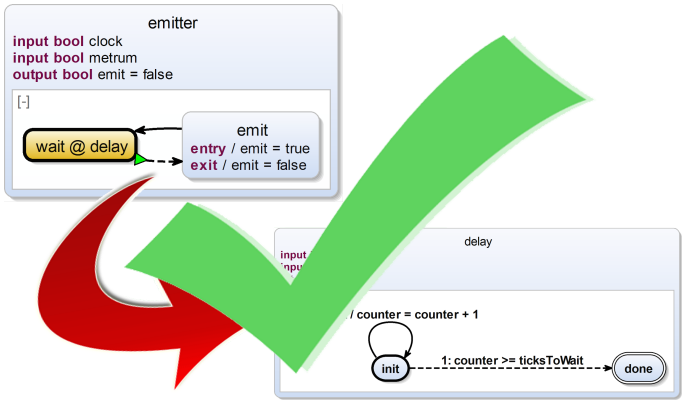
SCCharts Extensions - Roadmap

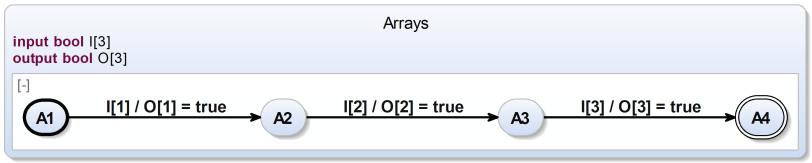




Category	Feature	OSem	Rail	Status
Core	Referenced SCCharts	✓	✓	
	Map & For	✓		
Declaration	Arrays	✓	✓	
	Const	✓		
	Extern			
	Structs	~		
Hostcode	Function calls	✓		

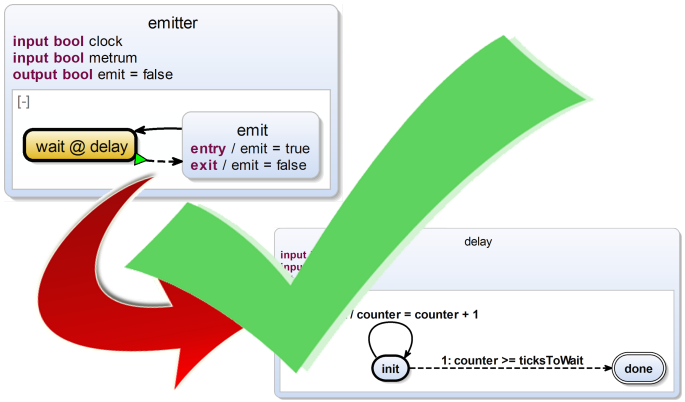
SCCharts Extensions - Roadmap

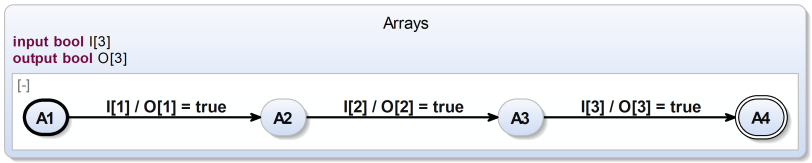




Category	Feature	OSem	Rail	Status
Core	Referenced SCCharts	✗	✗	✓
	Map & For	✗		
Declaration	Arrays	✗	✗	
	Const	✗		
	Extern			
	Structs	~		
Hostcode	Function calls	✗		

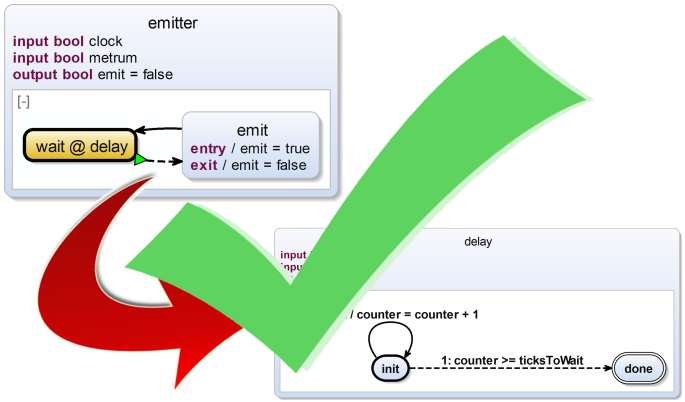
SCCharts Extensions - Roadmap

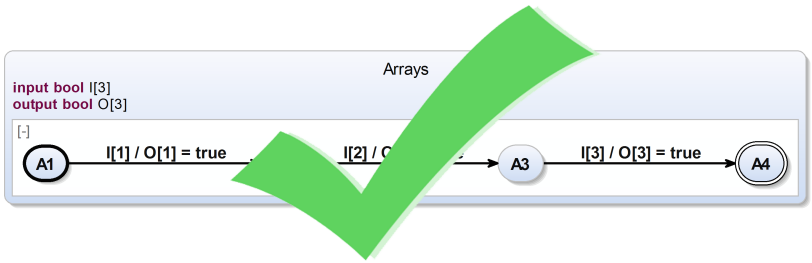




Category	Feature	OSem	Rail	Status
Core	Referenced SCCharts	✗	✗	✓
	Map & For	✗		~
Declaration	Arrays	✗	✗	
	Const	✗		
	Extern			
	Structs	~		
Hostcode	Function calls	✗		

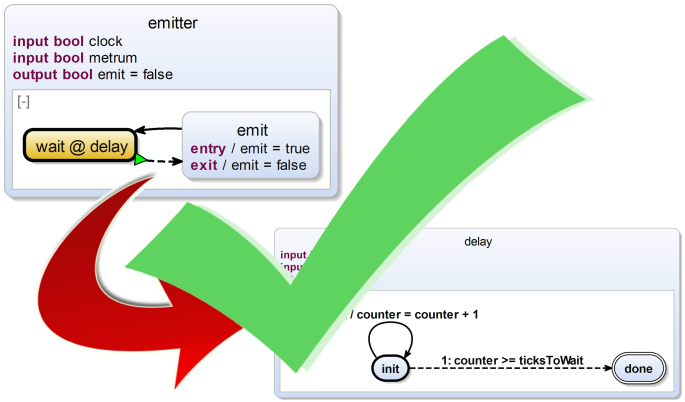
SCCharts Extensions - Roadmap

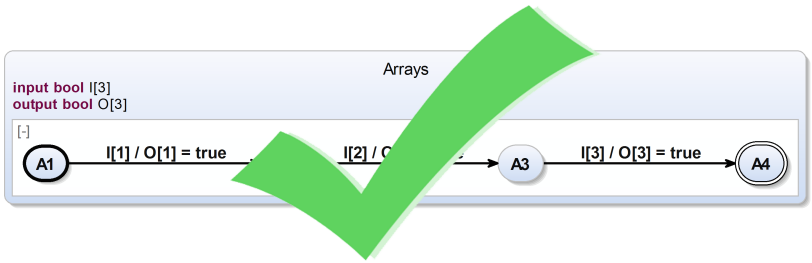




Category	Feature	OSem	Rail	Status
Core	Referenced SCCharts	✗	✗	✓
	Map & For	✗		~
Declaration	Arrays	✗	✗	✓
	Const	✗		
	Extern			
	Structs	~		
Hostcode	Function calls	✗		

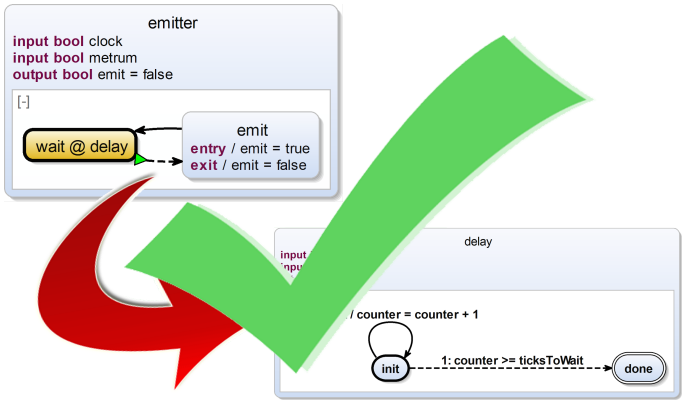
SCCharts Extensions - Roadmap

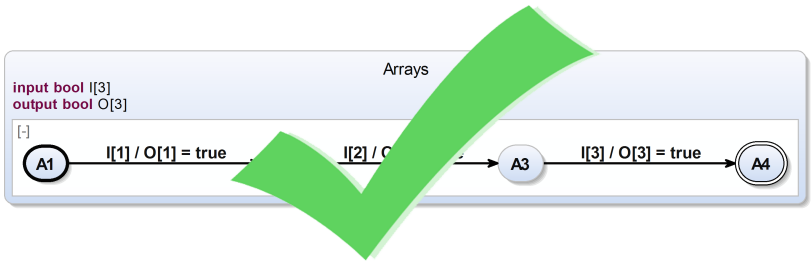




Category	Feature	OSem	Rail	Status
Core	Referenced SCCharts	✗	✗	✓
	Map & For	✗		~
Declaration	Arrays	✗	✗	✓
	Const	✗		✓
	Extern			
	Structs	~		
Hostcode	Function calls	✗		

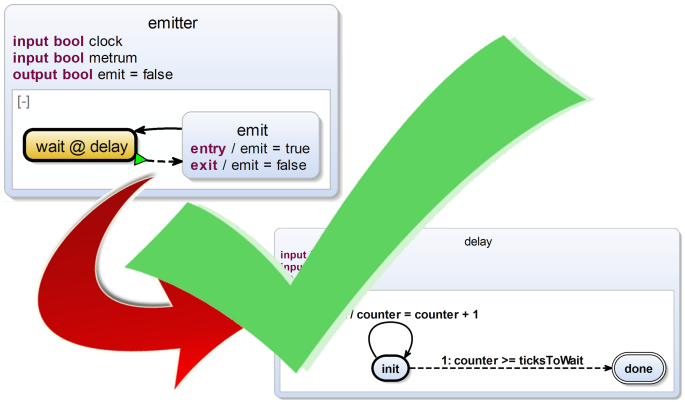
SCCharts Extensions - Roadmap

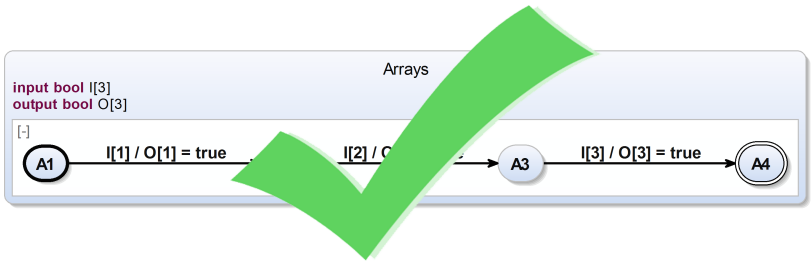




Category	Feature	OSem	Rail	Status
Core	Referenced SCCharts	✗	✗	✓
	Map & For	✗		~
Declaration	Arrays	✗	✗	✓
	Const	✗		✓
	Extern			✓
	Structs	~		
Hostcode	Function calls	✗		

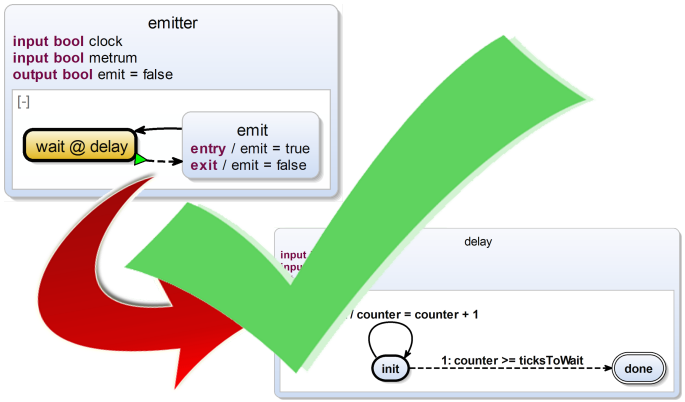
SCCharts Extensions - Roadmap

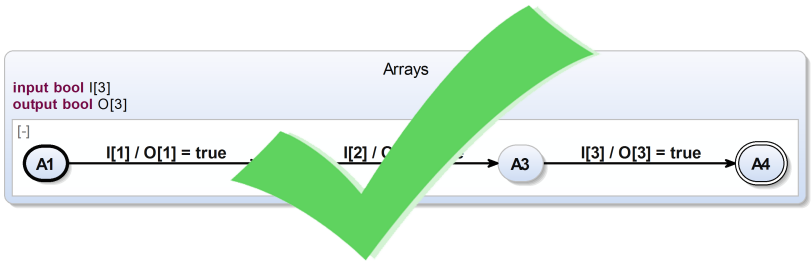




Category	Feature	OSem	Rail	Status
Core	Referenced SCCharts	✗	✗	✓
	Map & For	✗		~
Declaration	Arrays	✗	✗	✓
	Const	✗		✓
	Extern			✓
	Structs	~		~
Hostcode	Function calls	✗		

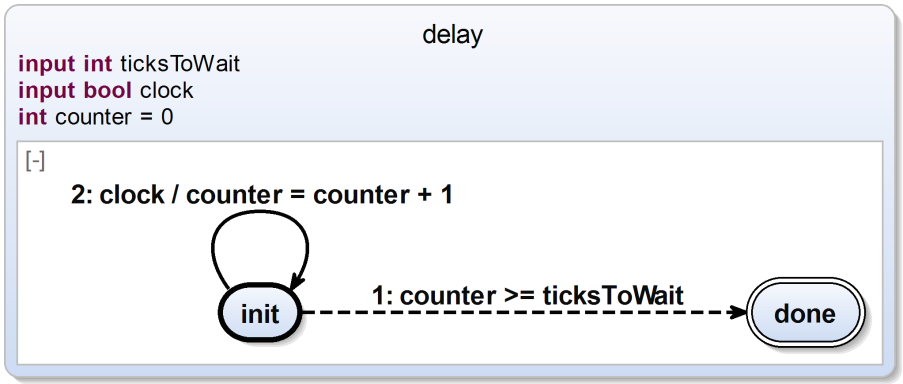
SCCharts Extensions - Roadmap





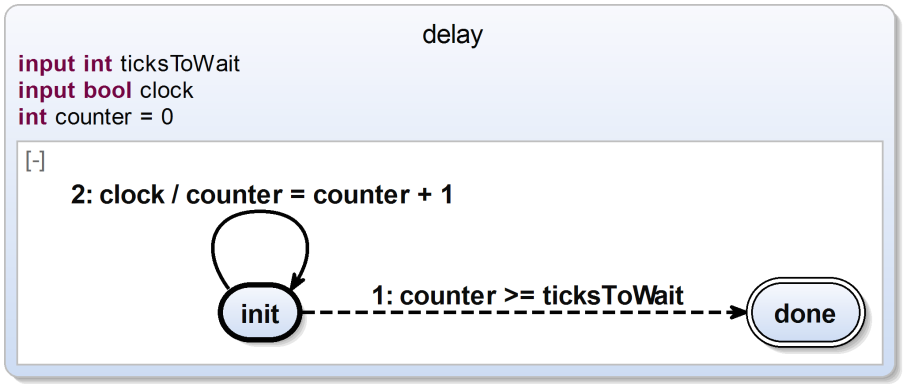
Category	Feature	OSem	Rail	Status
Core	Referenced SCCharts	✗	✗	✓
	Map & For	✗		~
Declaration	Arrays	✗	✗	✓
	Const	✗		✓
	Extern			✓
	Structs	~		~
Hostcode	Function calls	✗		✓

SCCharts Extensions - Referenced SCCharts

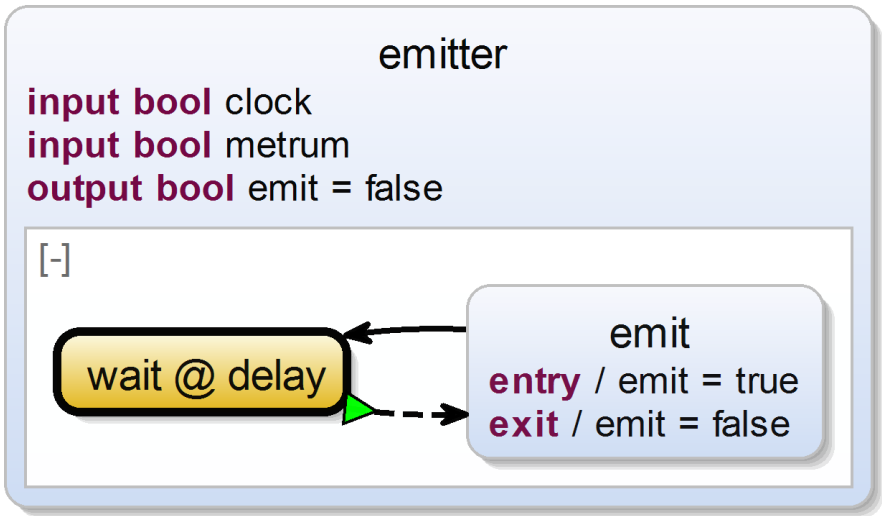



```
1  scchart delay {
2    input int ticksToWait;
3    input bool clock;
4    int counter = 0;
5
6    initial state init
7    --> done immediate with
8      counter >= ticksToWait
9    --> init with
10     clock / counter = counter + 1;
11
12   final state done;
13 }
```

SCCharts Extensions - Referenced SCCharts

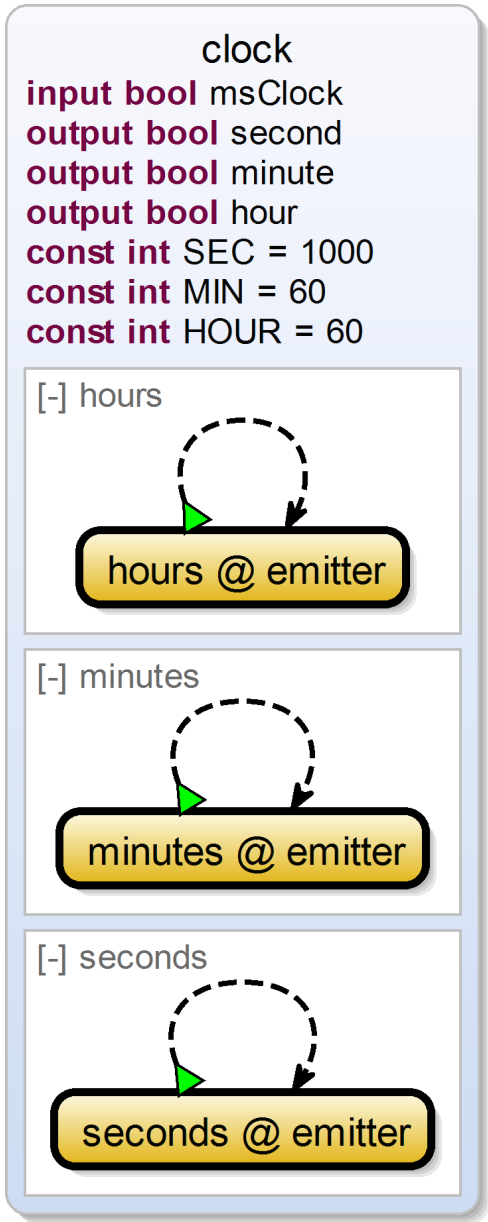



```
1  scchart delay {
2    input int ticksToWait;
3    input bool clock;
4    int counter = 0;
5
6    initial state init
7    --> done immediate with
8      counter >= ticksToWait
9    --> init with
10     clock / counter = counter + 1;
11
12   final state done;
13 }
```



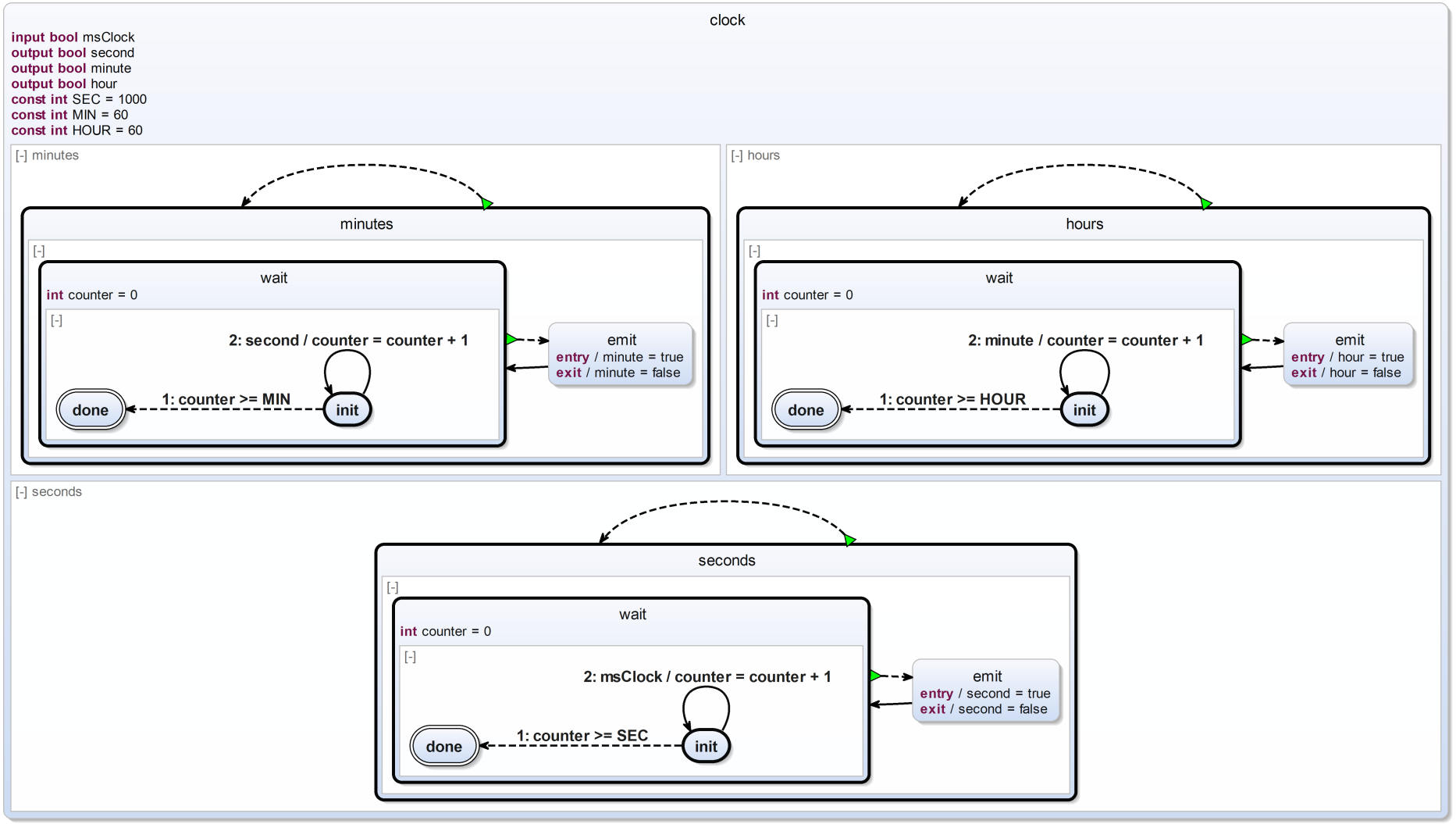

```
1  scchart emitter {
2    input bool clock;
3    input bool metrum;
4    output bool emit = false;
5
6    initial state wait
7    references delay
8    bind clock to clock,
9         ticksToWait to metrum
10   >-> emit;
11
12   state emit {
13     entry / emit = true;
14     exit / emit = false;
15   }
16   --> wait;
17 }
```

SCCharts Extensions - Referenced SCCharts



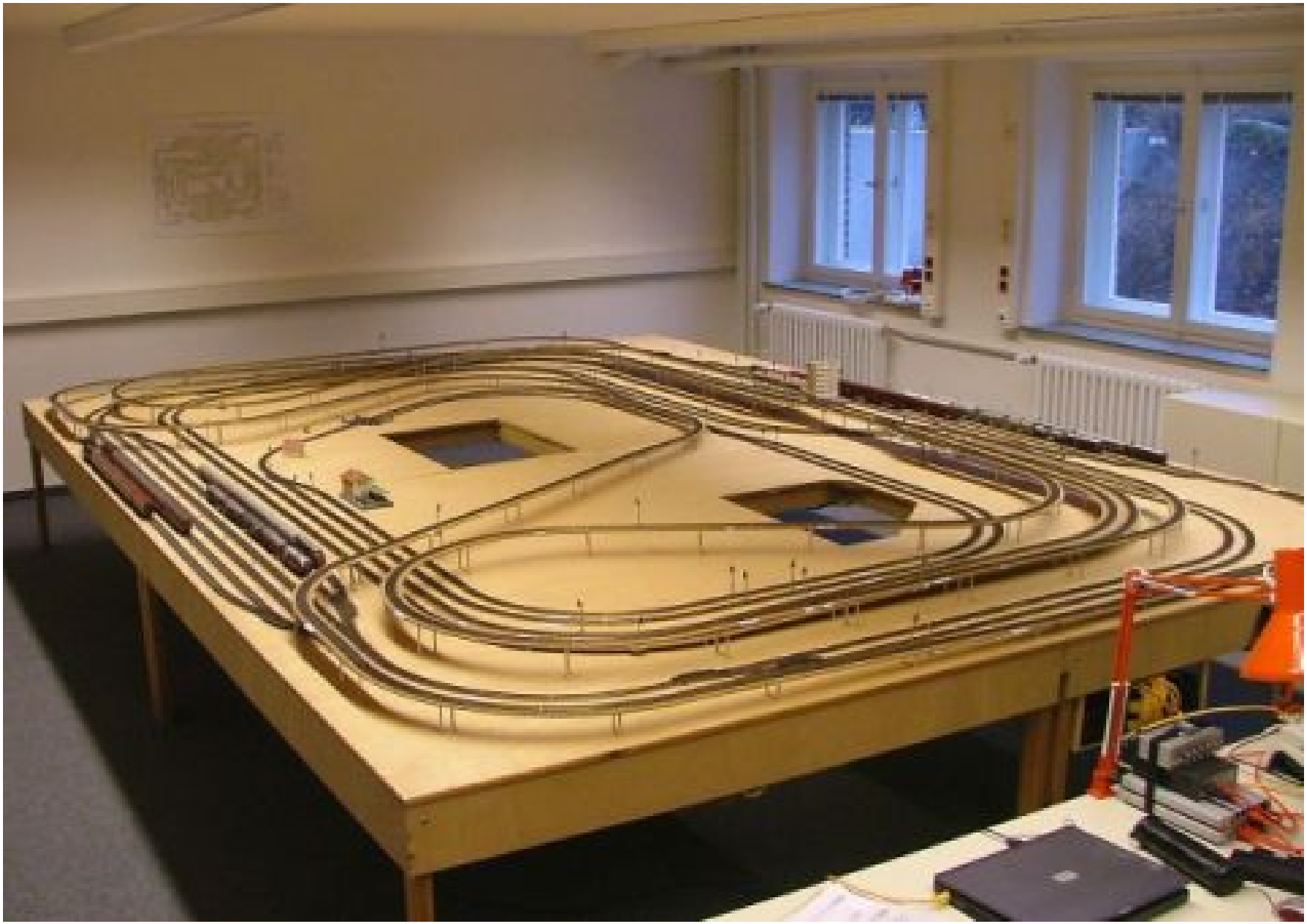

```
1  scchart clock {
2    input bool msClock;
3    output bool second, minute, hour;
4    const int SEC = 1000, MIN = 60,
5              HOUR = 60;
6
7    region seconds:
8
9    initial state seconds
10   references emitter
11   bind clock to msClock,
12        metrum to SEC,
13        emit to second
14   >-> seconds;
15
16   region minutes:
17
18   initial state minutes
19   references emitter
20   bind clock to second,
21        metrum to MIN,
22        emit to minute
23   >-> minutes;
24
25   region hours:
26
27   initial state hours
28   references emitter
29   bind clock to minute,
30        metrum to HOUR,
31        emit to hour
32   >-> hours;
33 }
```

SCCharts Extensions - Referenced SCCharts



Overview

Project Overview

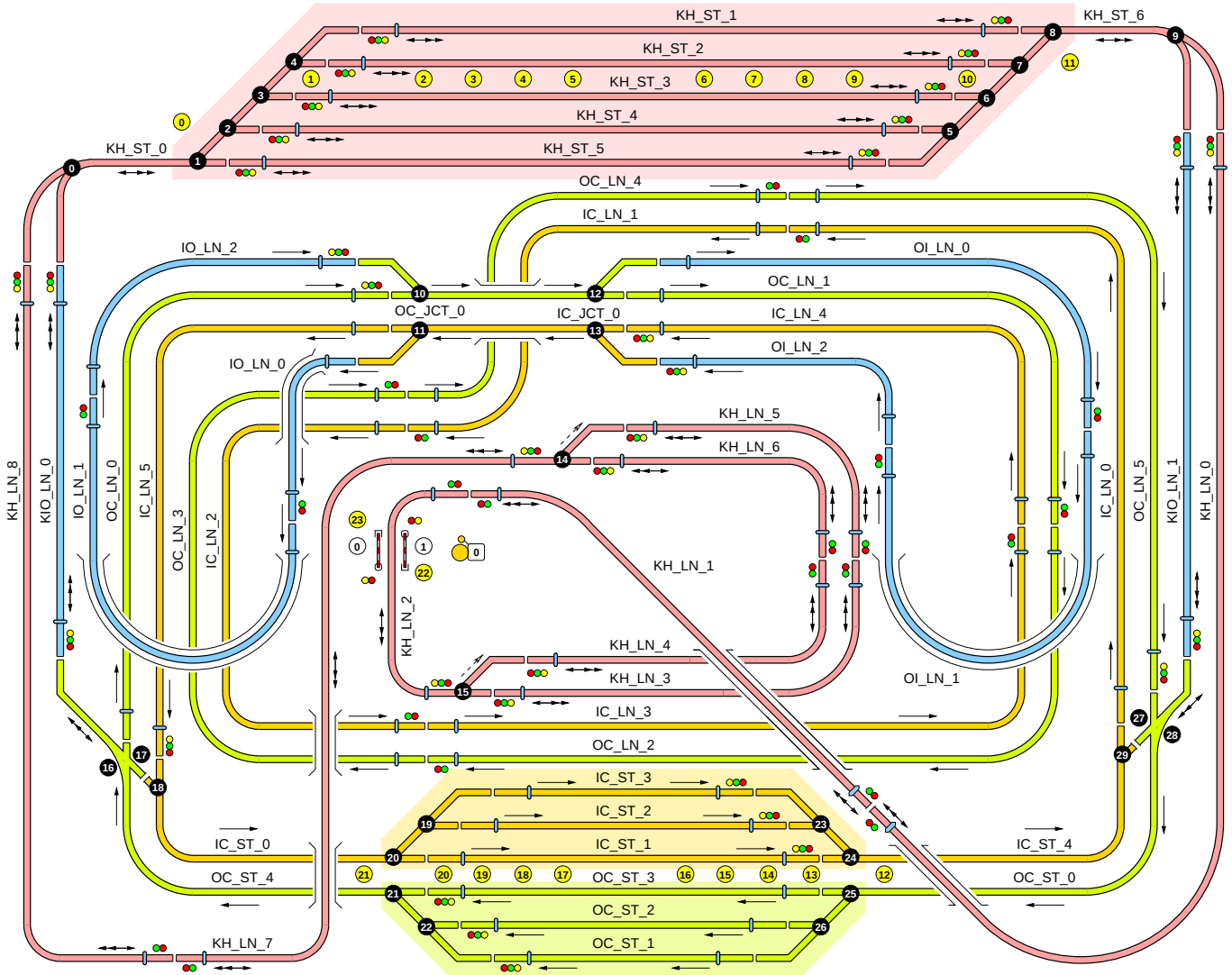


Project Overview

Project Overview

Model Railway Track Layout

Realtime and Embedded Systems Group



Symbols

Track segments

- Inner Circle
- Outer Circle
- Kicking Horse Pass
- Interconnections

Track specialties

- Bridge
- Point or crossing
- Railroad crossing

Directions

- Unidirectional block
- Bidirectional block with forward direction
- Preferred direction

Electronics

- Block isolation
- Reed contact
- Block signal
- Lighting
- Point operating unit

Stephan Höhrmann, January 2006
http://www.informatik.uni-kiel.de/~railway/

Project Overview - Controller Size

Approximatly...

Project Overview - Controller Size

Approximatly...

- 135.000 states

Project Overview - Controller Size

Approximatly...

- 135.000 states
- 17.000 regions

Project Overview - Controller Size

Approximatly...

- 135.000 states
- 17.000 regions
- 152.000 transitions

Project Overview - Controller Size

Approximatly...

- 135.000 states
- 17.000 regions
- 152.000 transitions
- 127.000 SCG nodes

Project Overview - Controller Size

Approximatly...

- 135.000 states
- 17.000 regions
- 152.000 transitions

- 127.000 SCG nodes
- 256.000 dependencies
  - in 50.000 nodes

Project Overview - Controller Size

Approximatly...

- 135.000 states
- 17.000 regions
- 152.000 transitions

- 127.000 SCG nodes
- 256.000 dependencies
  - in 50.000 nodes
- 88.000 scheduling blocks

Project Overview - Controller Size

Approximatly...

- 135.000 states
- 17.000 regions
- 152.000 transitions

- 127.000 SCG nodes
- 256.000 dependencies
  - in 50.000 nodes
- 88.000 scheduling blocks
- 1.400.000 lines of generated C code
  - still 650.000 lines after beautifier

Project Overview - Controller Size

Approximatly...

- 135.000 states
- 17.000 regions
- 152.000 transitions

- 127.000 SCG nodes
- 256.000 dependencies
  - in 50.000 nodes
- 88.000 scheduling blocks
- 1.400.000 lines of generated C code
  - still 650.000 lines after beautifier
- 3.000 lines of own C code

Project Overview - Controller Size

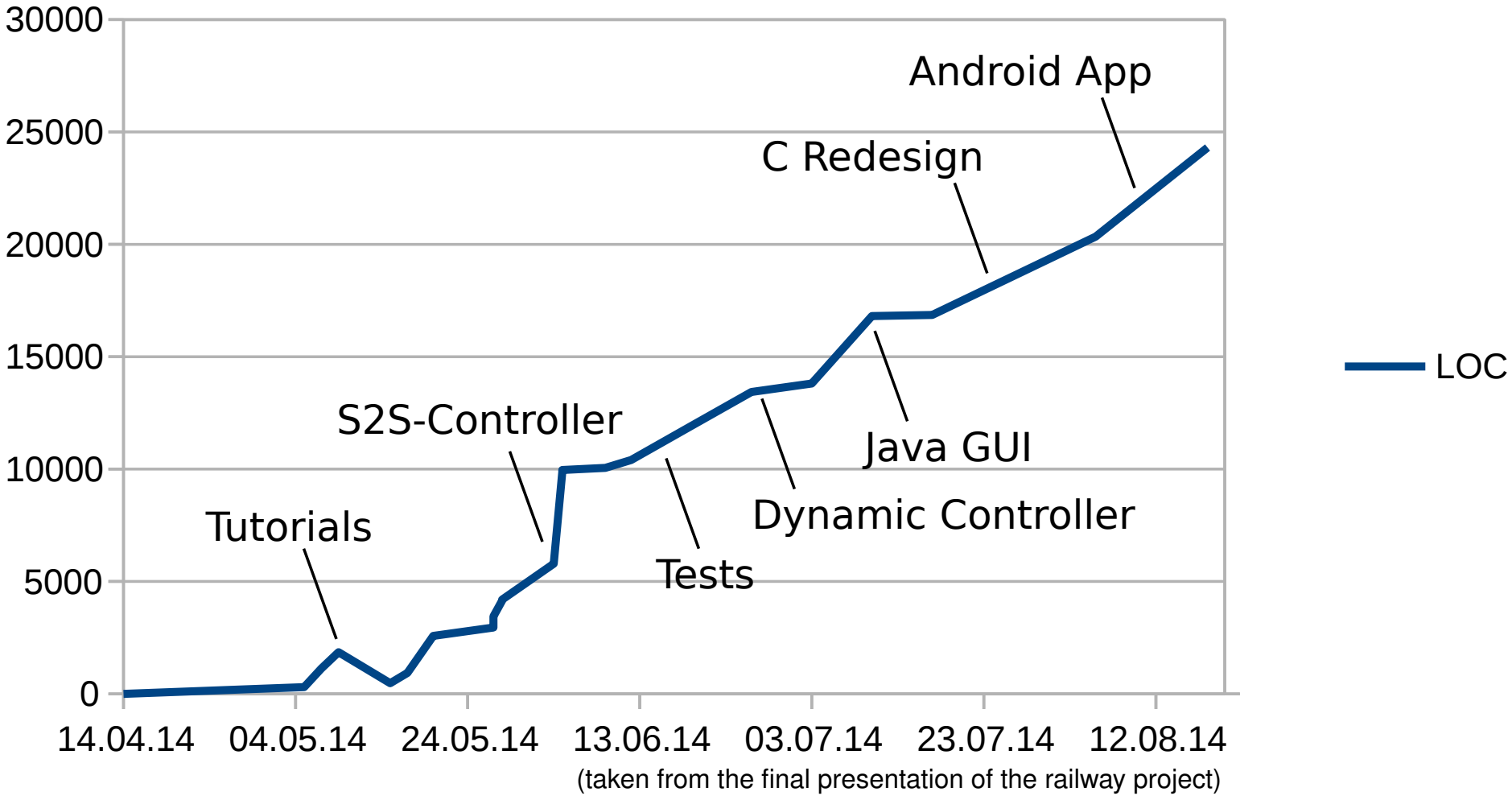
Approximatly...

- 135.000 states
- 17.000 regions
- 152.000 transitions

- 127.000 SCG nodes
- 256.000 dependencies
  - in 50.000 nodes
- 88.000 scheduling blocks
- 1.400.000 lines of generated C code
  - still 650.000 lines after beautifier
- 3.000 lines of own C code

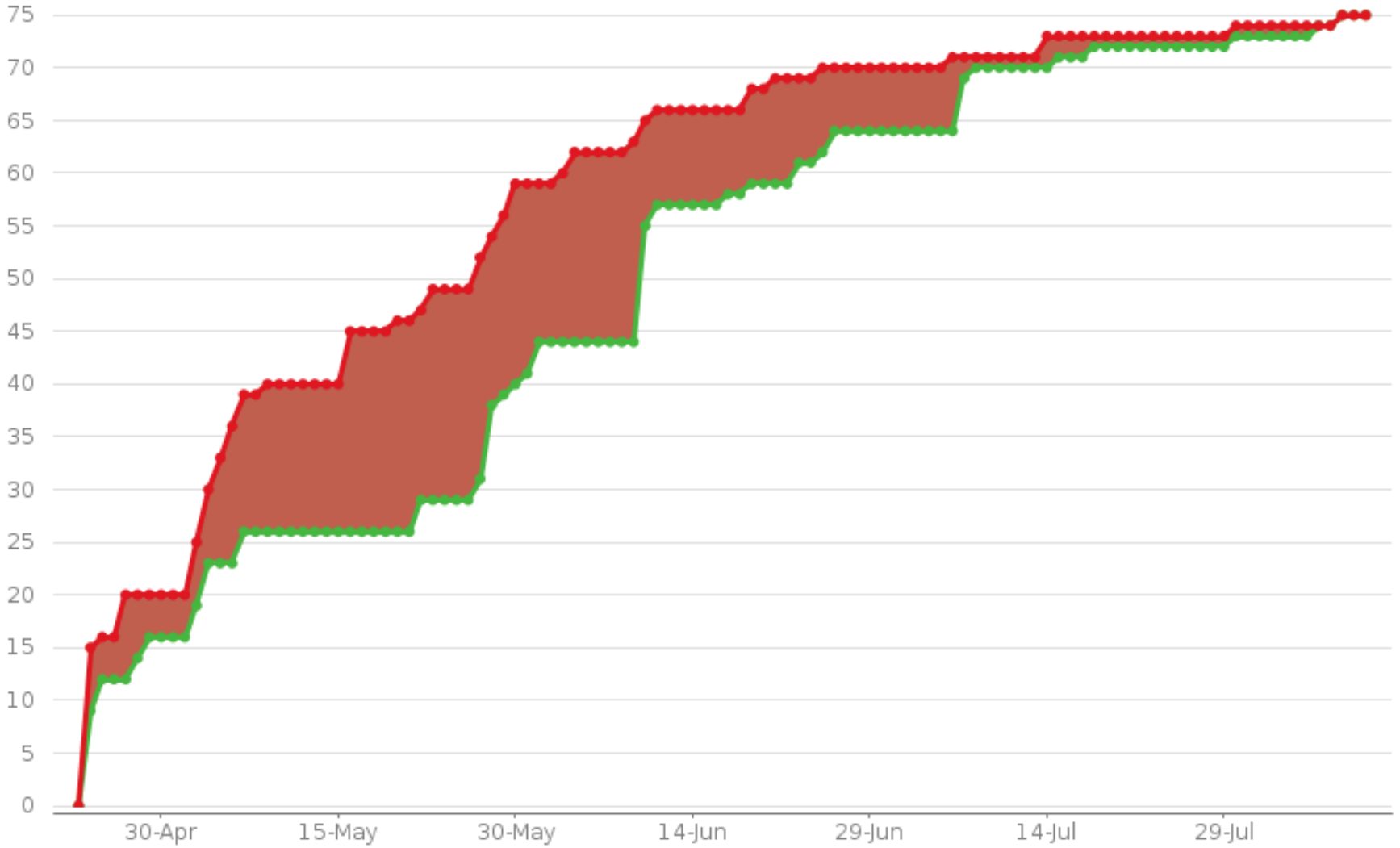
As comparison: The Wrist Watch has ~400 states.

Project Overview - Controller Size



(taken from the final presentation of the railway project)

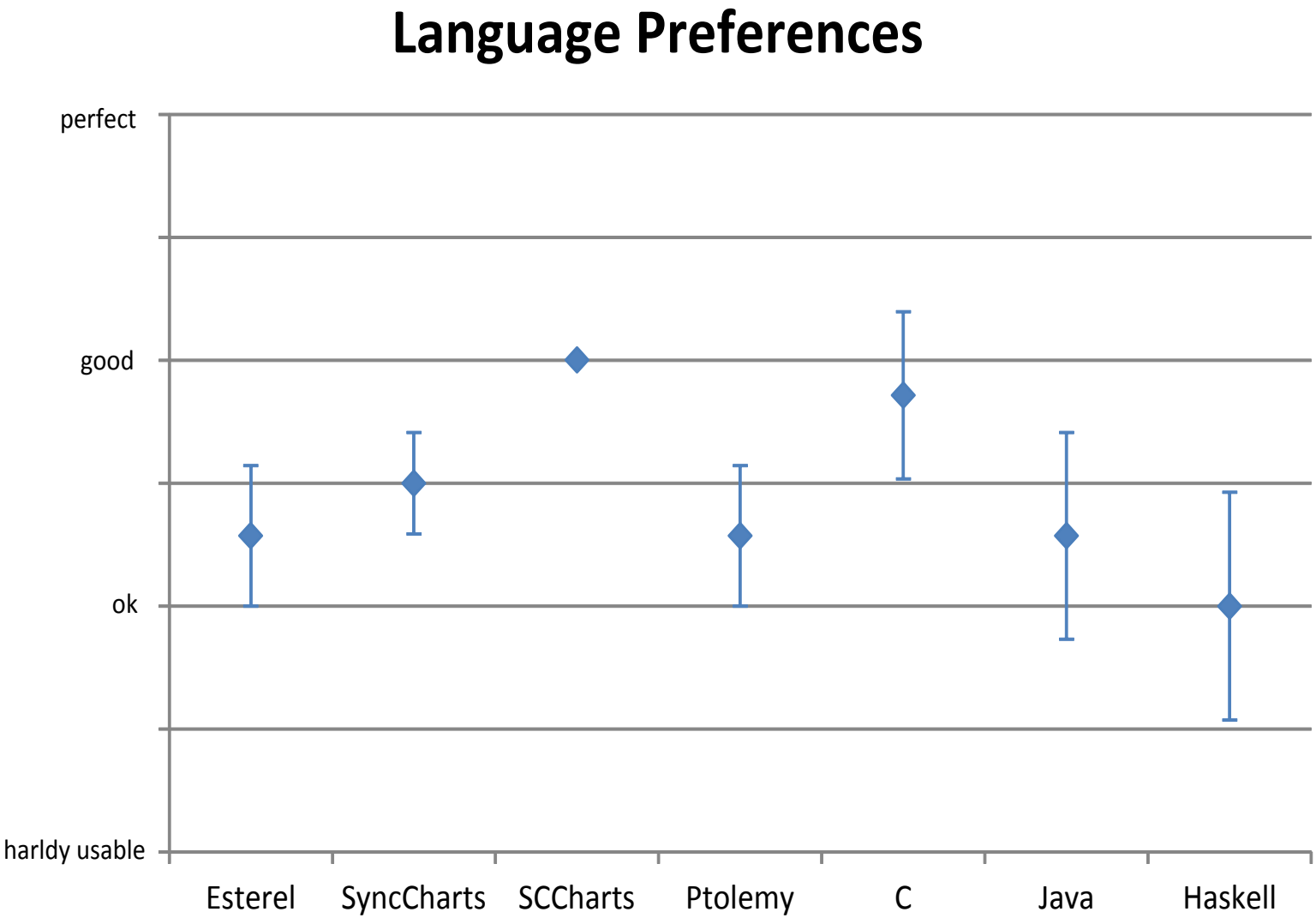
Project Overview - Joint Work



Overview

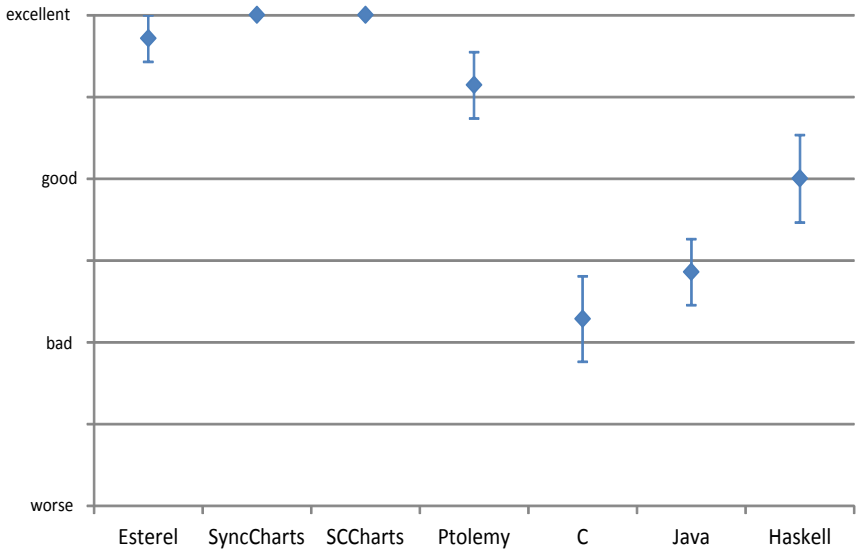
Language Evaluation

To which extent would you like to use the following modeling/programming languages for this project?

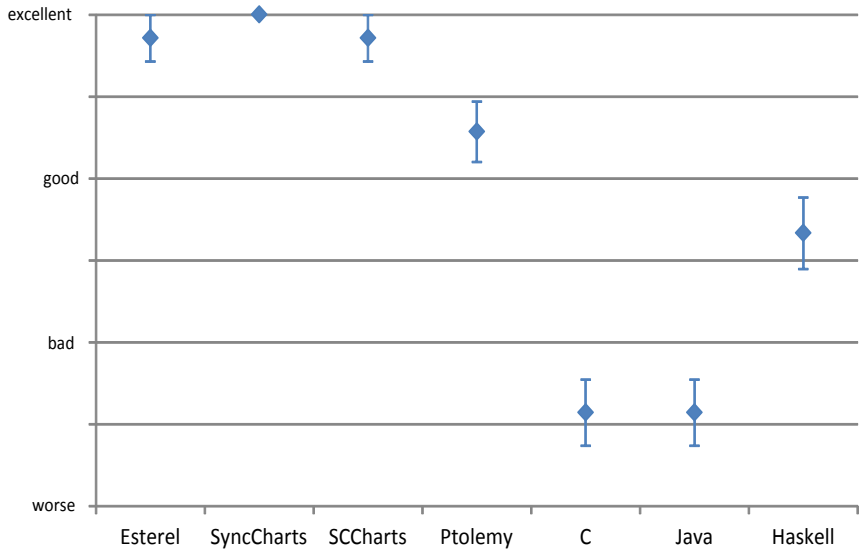


Language Evaluation

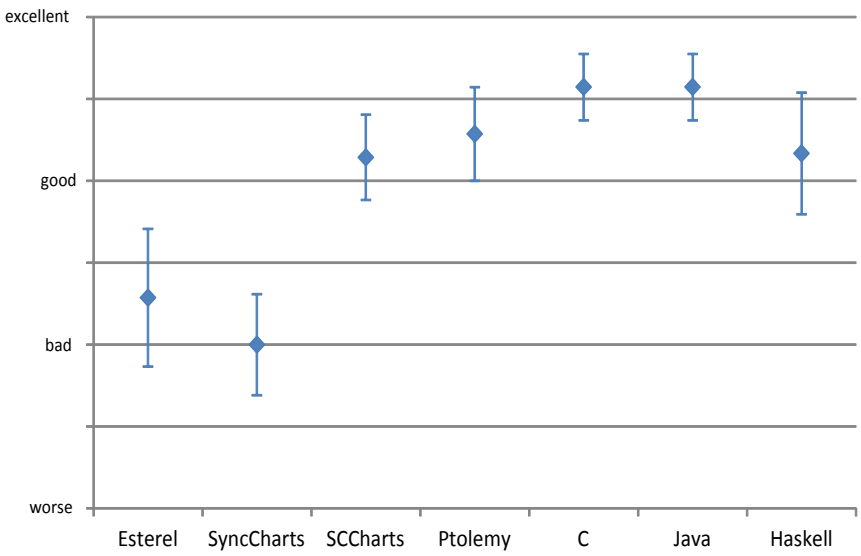
General determinism



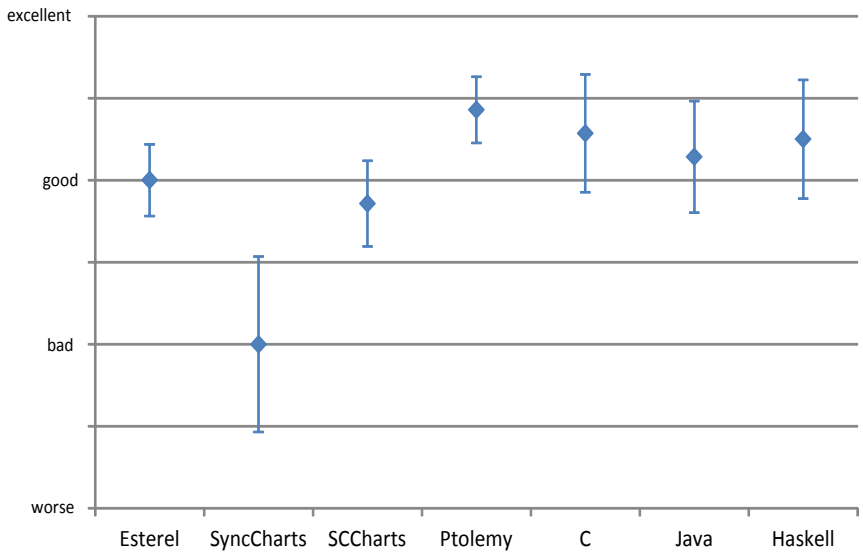
Deterministic concurrency



Sequentiality

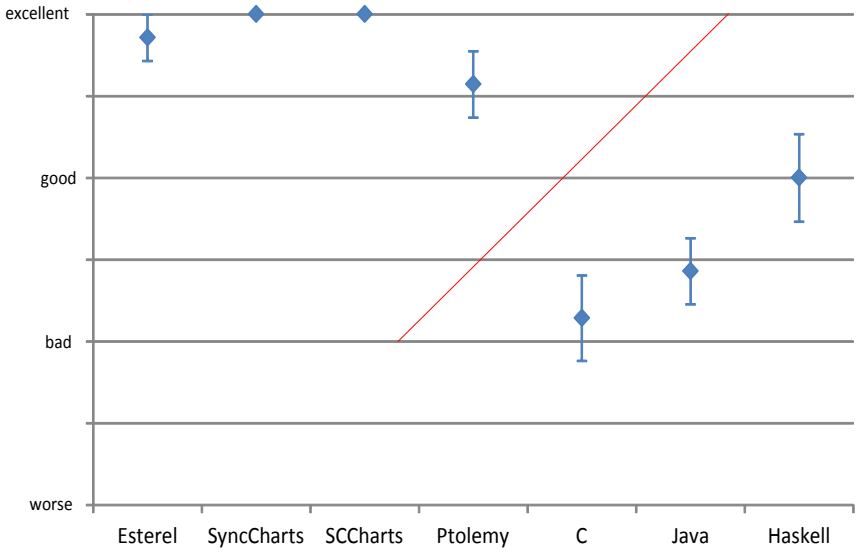


Composability

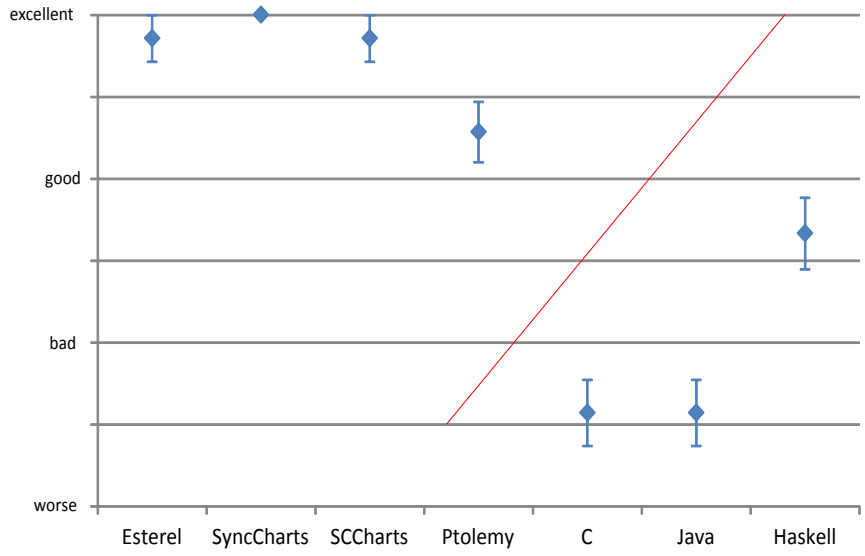


Language Evaluation

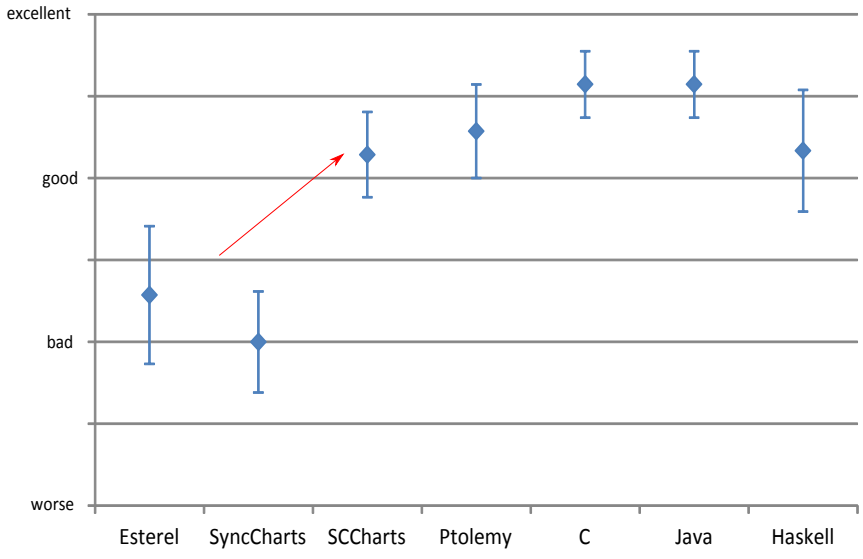
General determinism



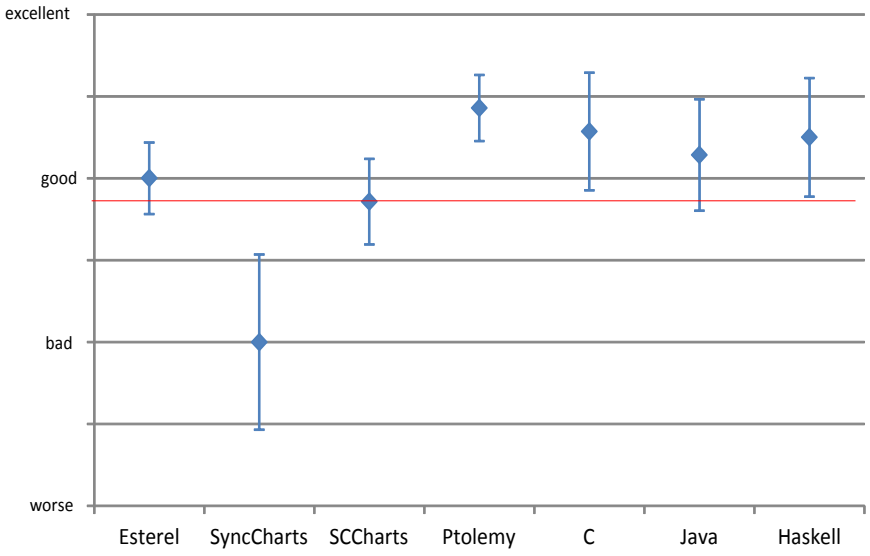
Deterministic concurrency



Sequentiality

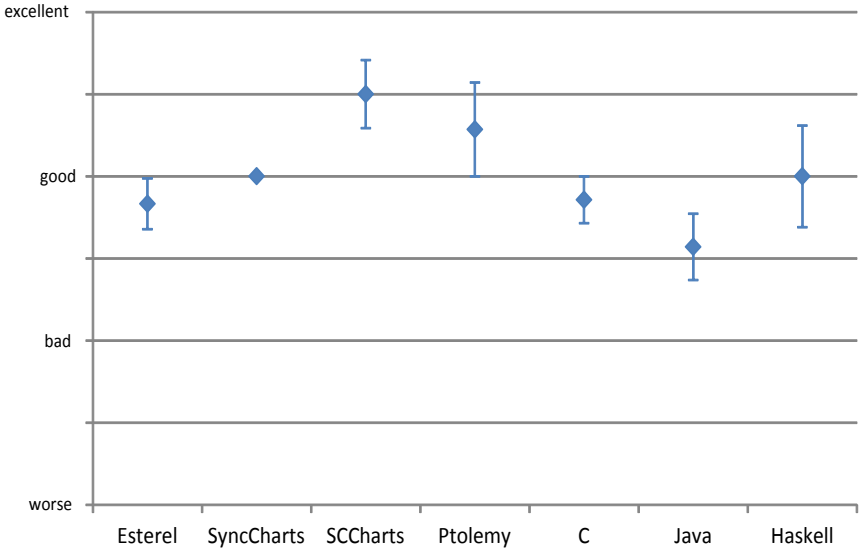


Composability

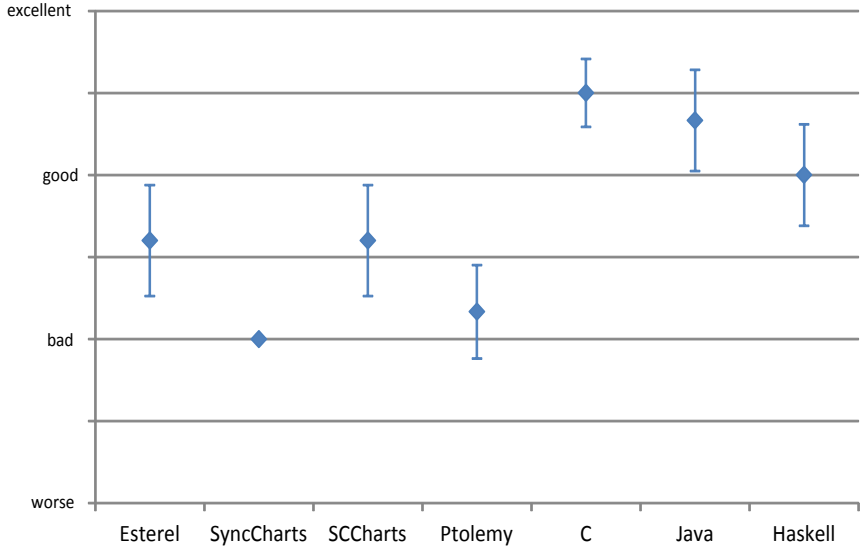


Language Evaluation

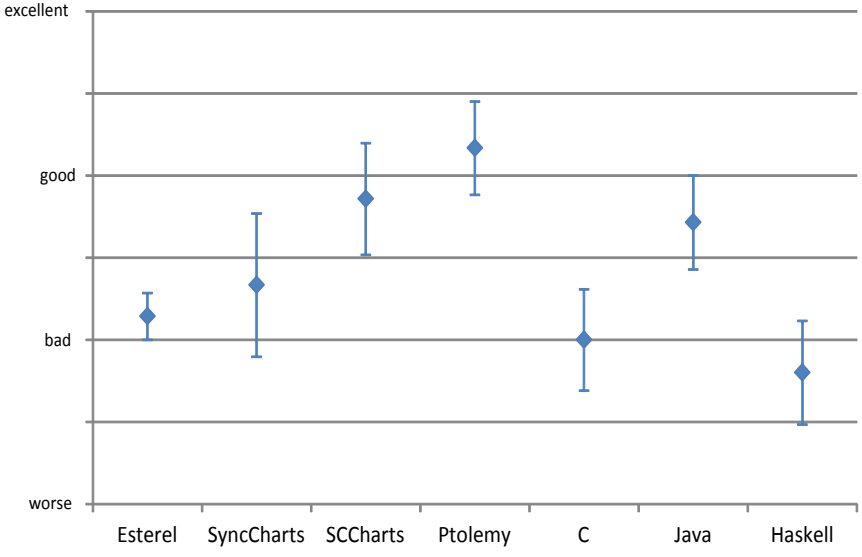
Solving abstract problems



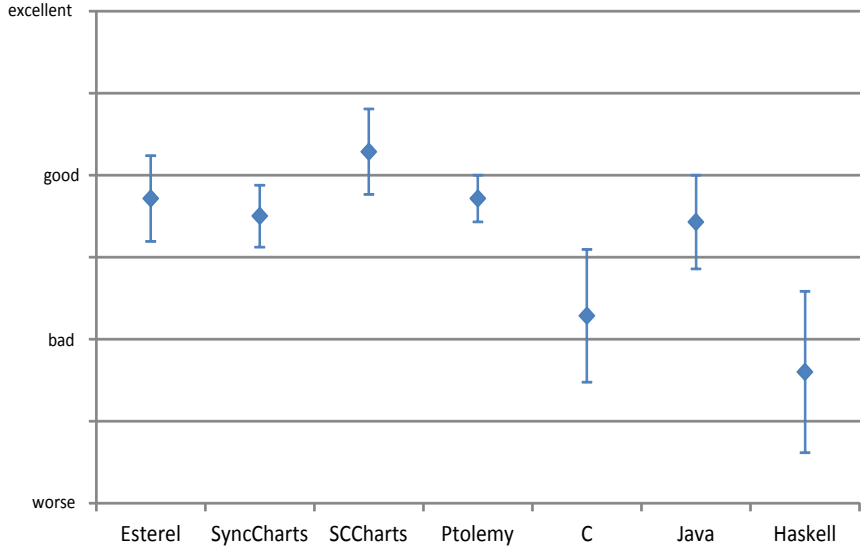
Solving low-level problems



Understandability

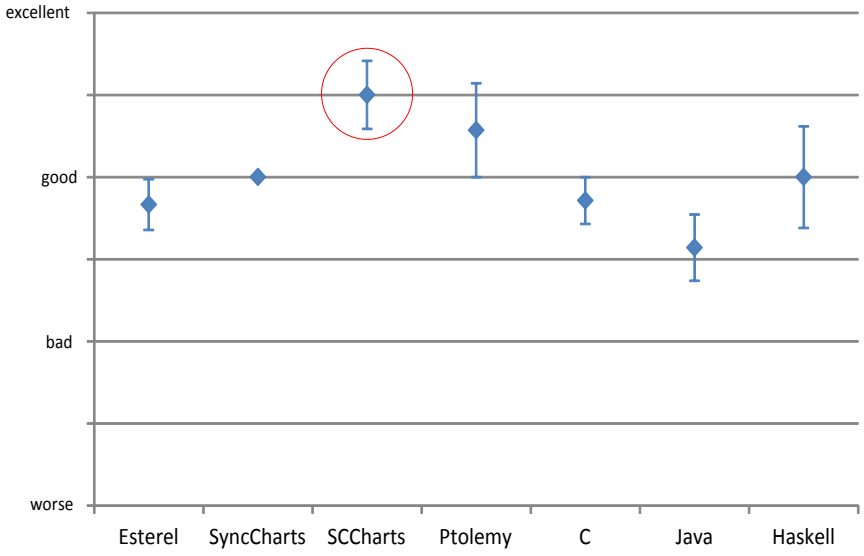


Simplicity

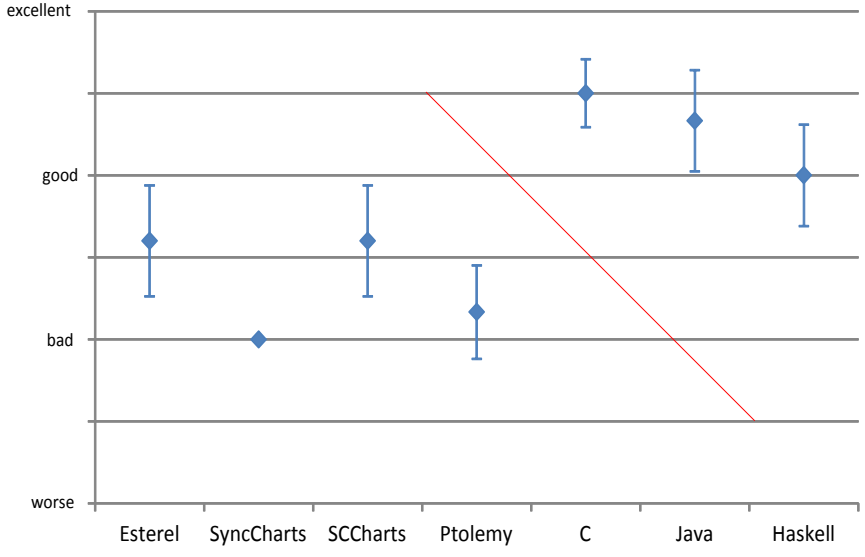


Language Evaluation

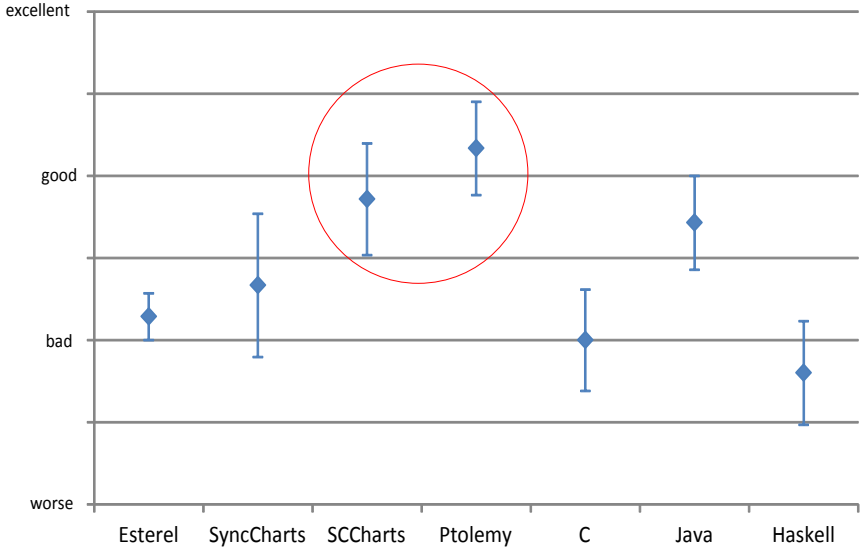
Solving abstract problems



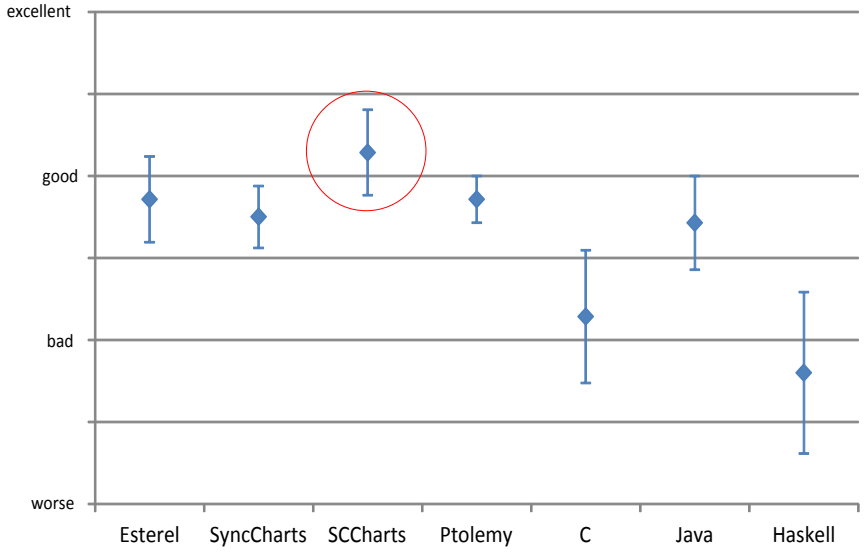
Solving low-level problems



Understandability

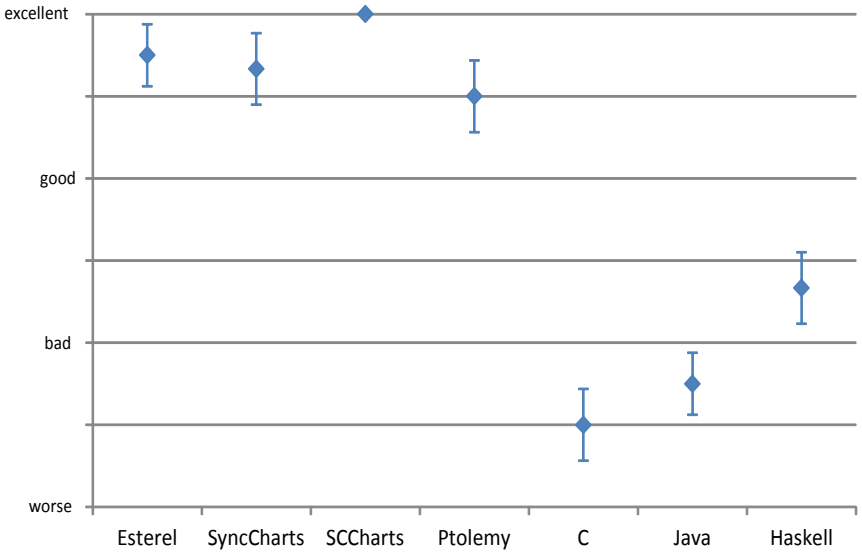


Simplicity

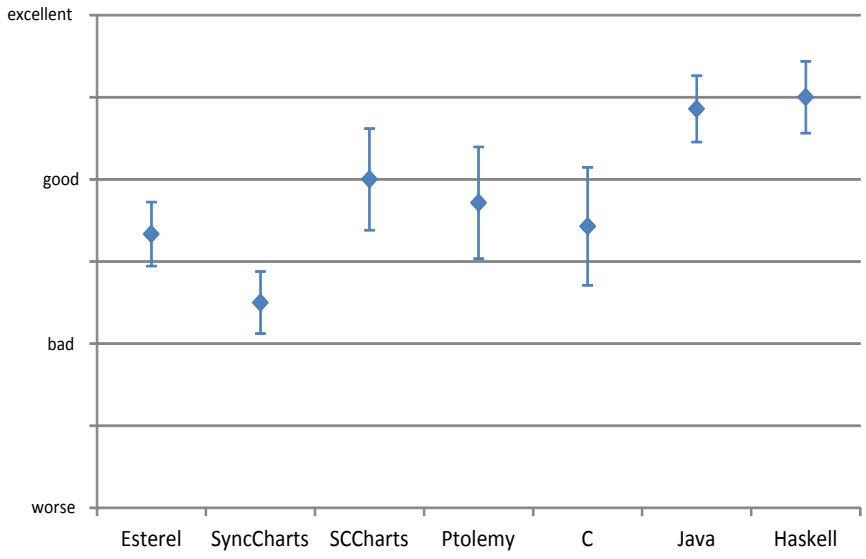


Language Evaluation

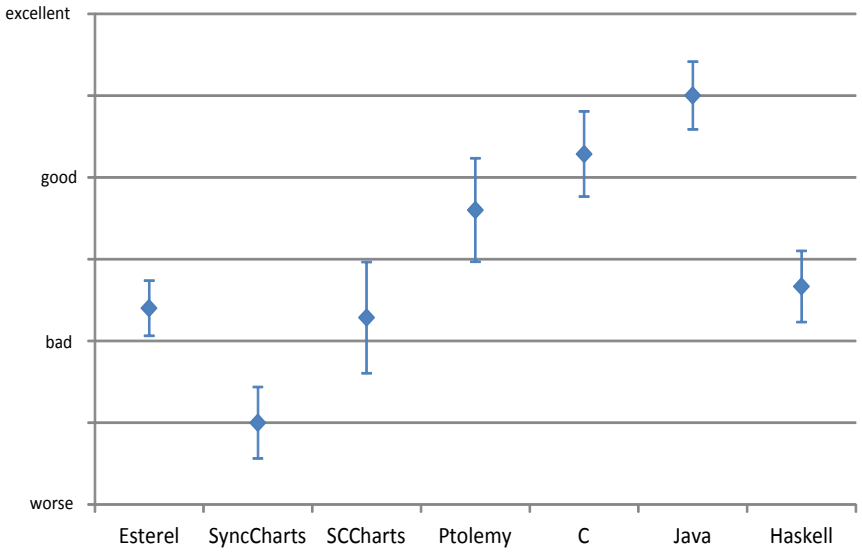
Separate Timing & Functionality



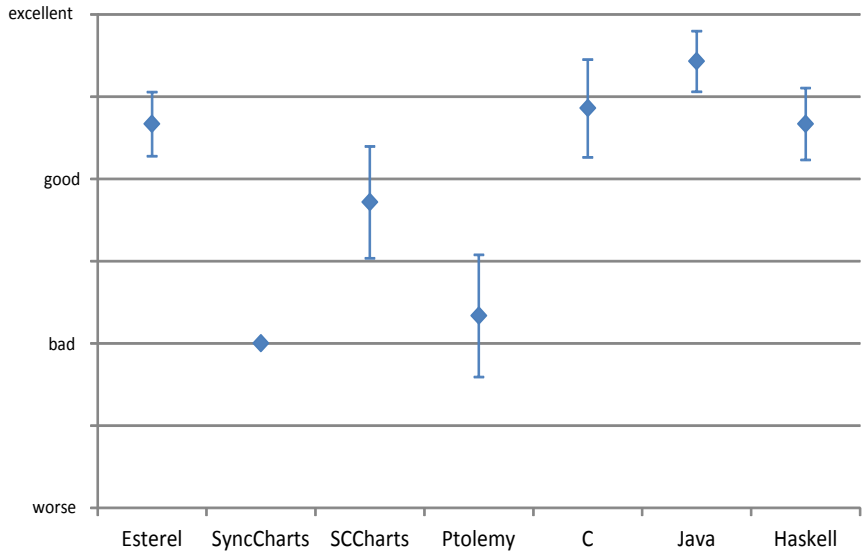
Maintainability



Debugging

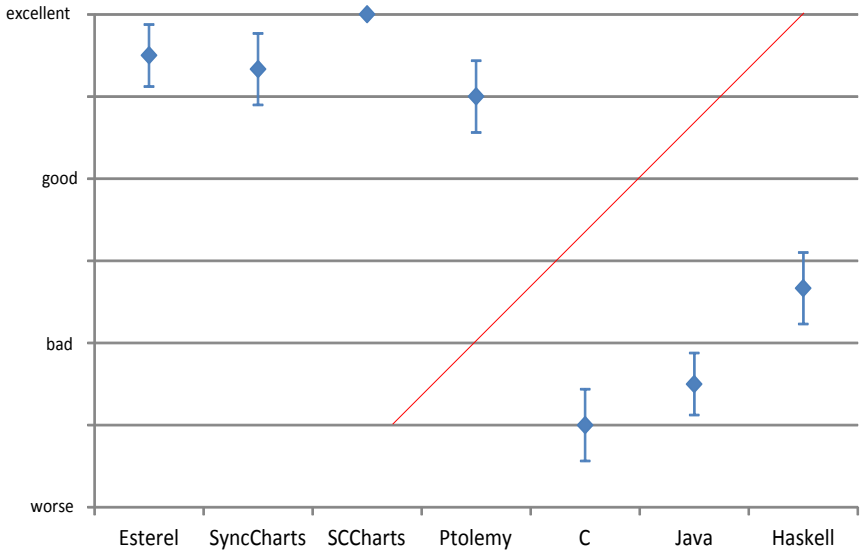


Team development

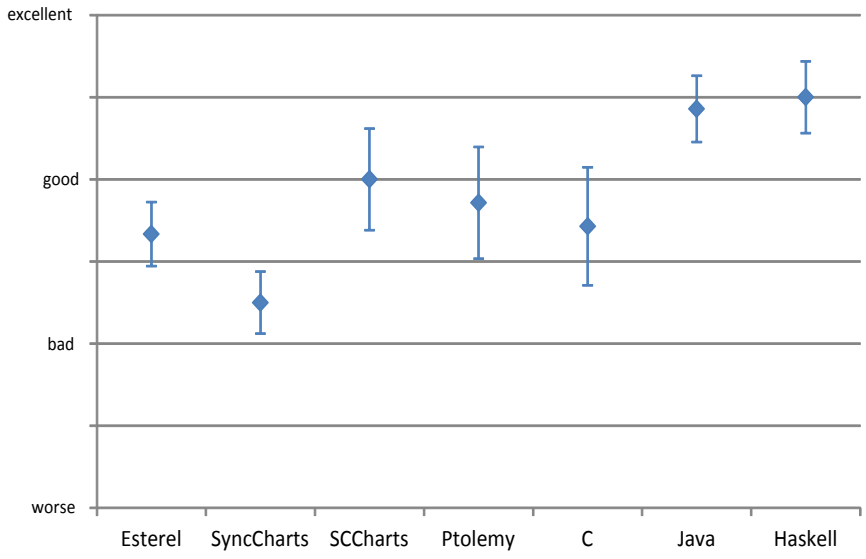


Language Evaluation

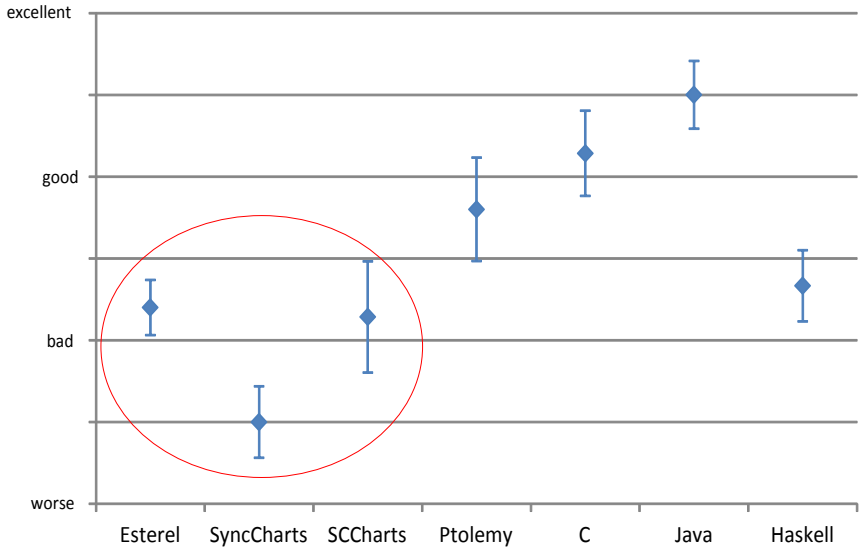
Separate Timing & Functionality



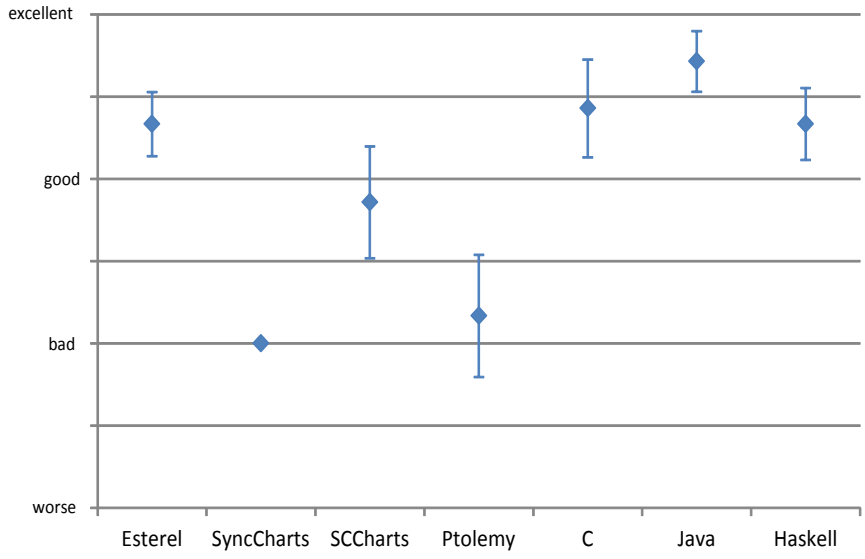
Maintainability



Debugging



Team development



Language Evaluation

What were the most challenging functionalities to be implemented?

Language Evaluation

What were the most challenging functionalities to be implemented?

- Dealing with cyclic dependencies x2

Language Evaluation

What were the most challenging functionalities to be implemented?

- Dealing with cyclic dependencies x2
- Mutual exclusion problems x3

Language Evaluation

What were the most challenging functionalities to be implemented?

- Dealing with cyclic dependencies x2
- Mutual exclusion problems x3
- Design and compiling problems x2

Language Evaluation

What were the most challenging functionalities to be implemented?

- Dealing with cyclic dependencies x2
- Mutual exclusion problems x3
- Design and compiling problems x2
- ... and feature specific problems

Language Evaluation

First conclusions...

Language Evaluation

First conclusions...

- Performs better than other synchroneous languages,

Language Evaluation

First conclusions...

- Performs better than other synchroneous languages,
- is understandable, simplistic and maintainable,

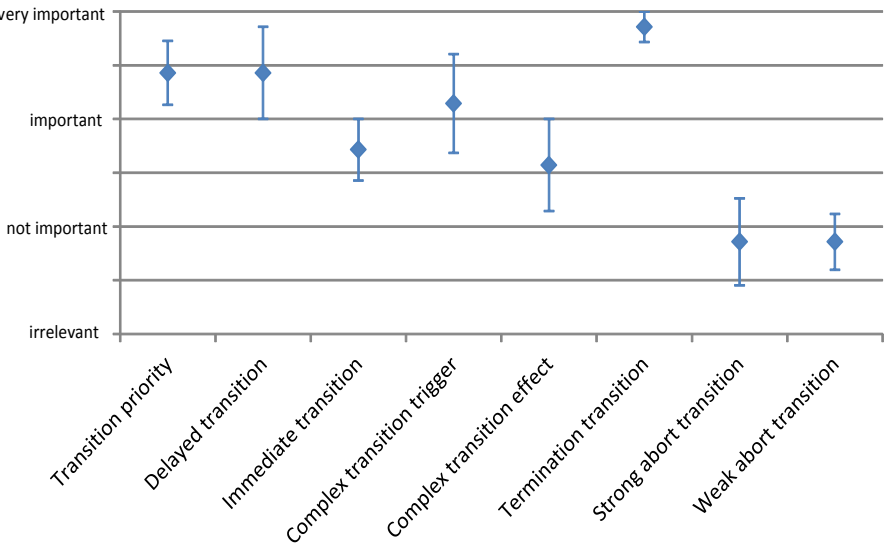
Language Evaluation

First conclusions...

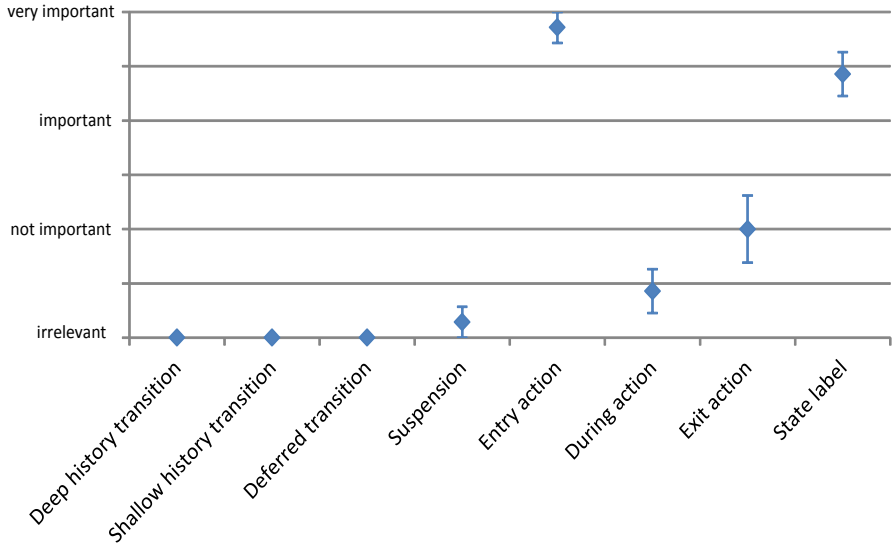
- Performs better than other synchroneous languages,
- is understandable, simplistic and maintainable,
- but still needs better support for debugging, composability and team development.

Language Evaluation - Features

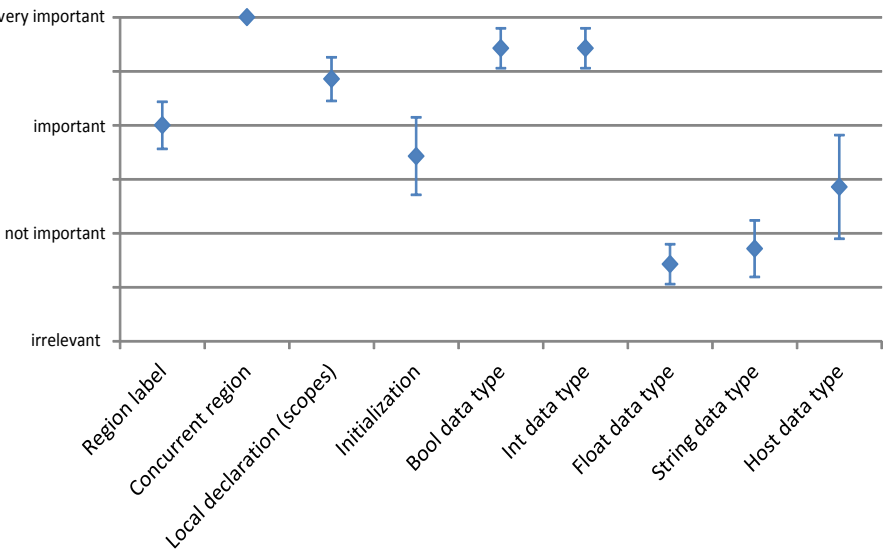
Extended features 1/4



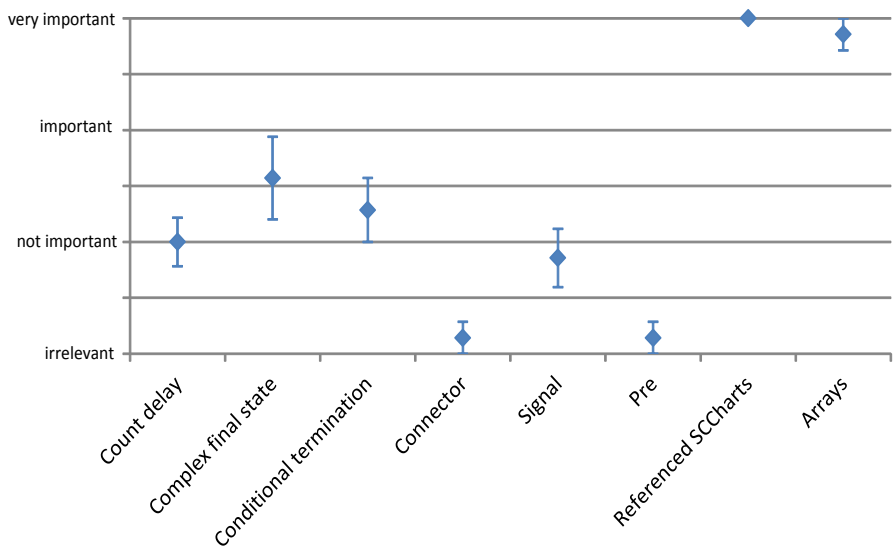
Extended features 2/4



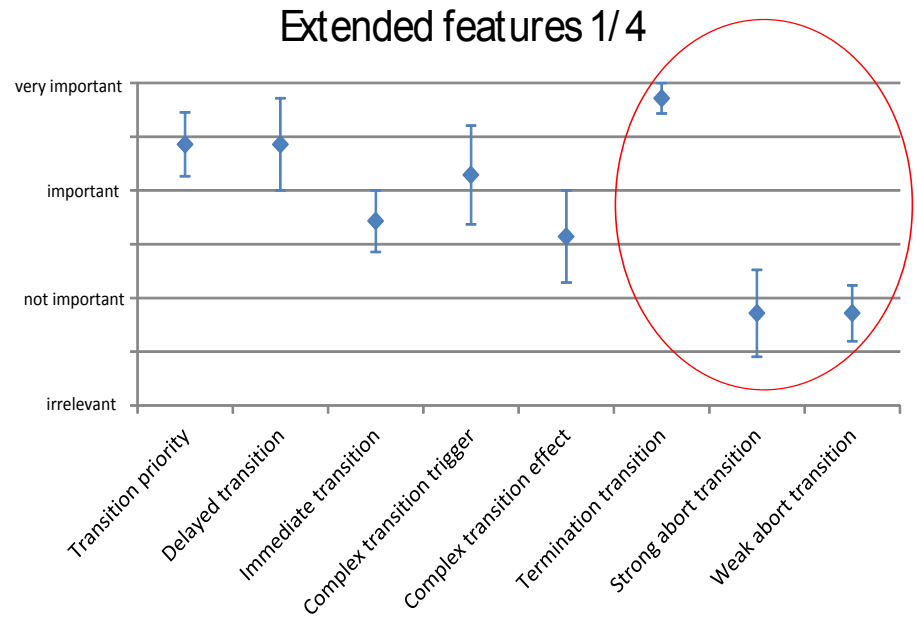
Extended features 3/4

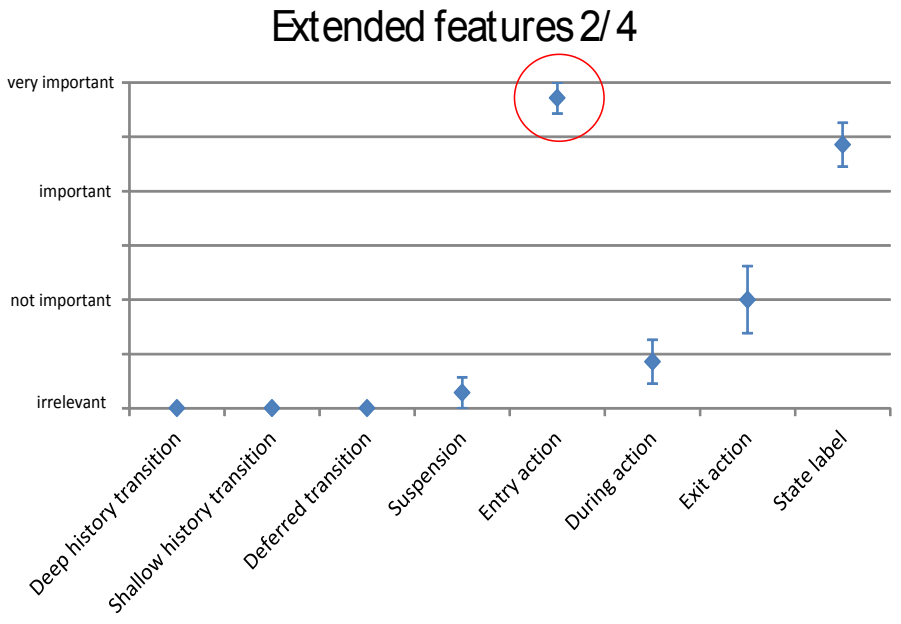


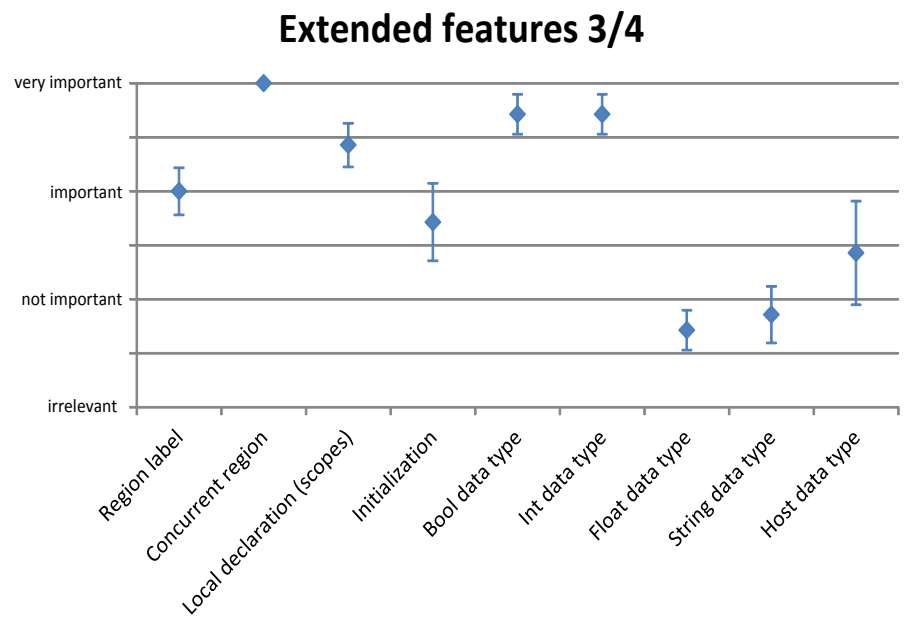
Extended features 4/4

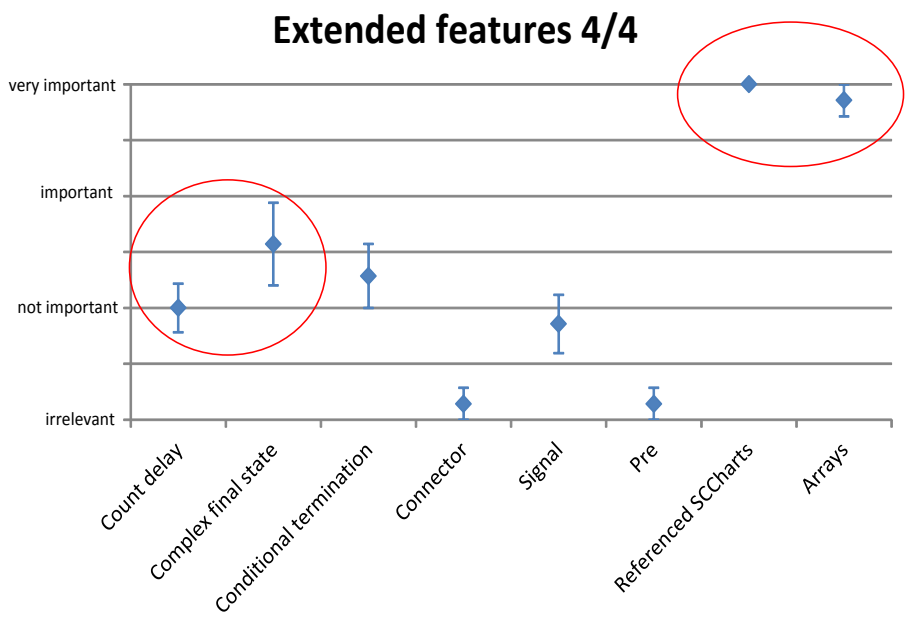


Language Evaluation - Features









Language Evaluation - Features

Which (extended) features did you miss?

Language Evaluation - Features

Which (extended) features did you miss?

- Generating multiple similar regions

Language Evaluation - Features

Which (extended) features did you miss?

- Generating multiple similar regions
  $\longrightarrow$ Map & For

Language Evaluation - Features

Which (extended) features did you miss?

- Generating multiple similar regions
  $\longrightarrow$ Map & For
- Broken features
  - complex final states x2
  - conditional termination
  - count delay

Language Evaluation - Features

Which (extended) features did you miss?

- Generating multiple similar regions
  $\longrightarrow$ Map & For
- Broken features
  - complex final states x2
  - conditional termination
  - count delay
    $\longrightarrow$ Fix it!

Language Evaluation - Features

Which (extended) features did you miss?

- Generating multiple similar regions
  $\longrightarrow$ Map & For
- Broken features
  - complex final states x2
  - conditional termination
  - count delay
    $\longrightarrow$ Fix it!
- None x3

Language Evaluation - Features

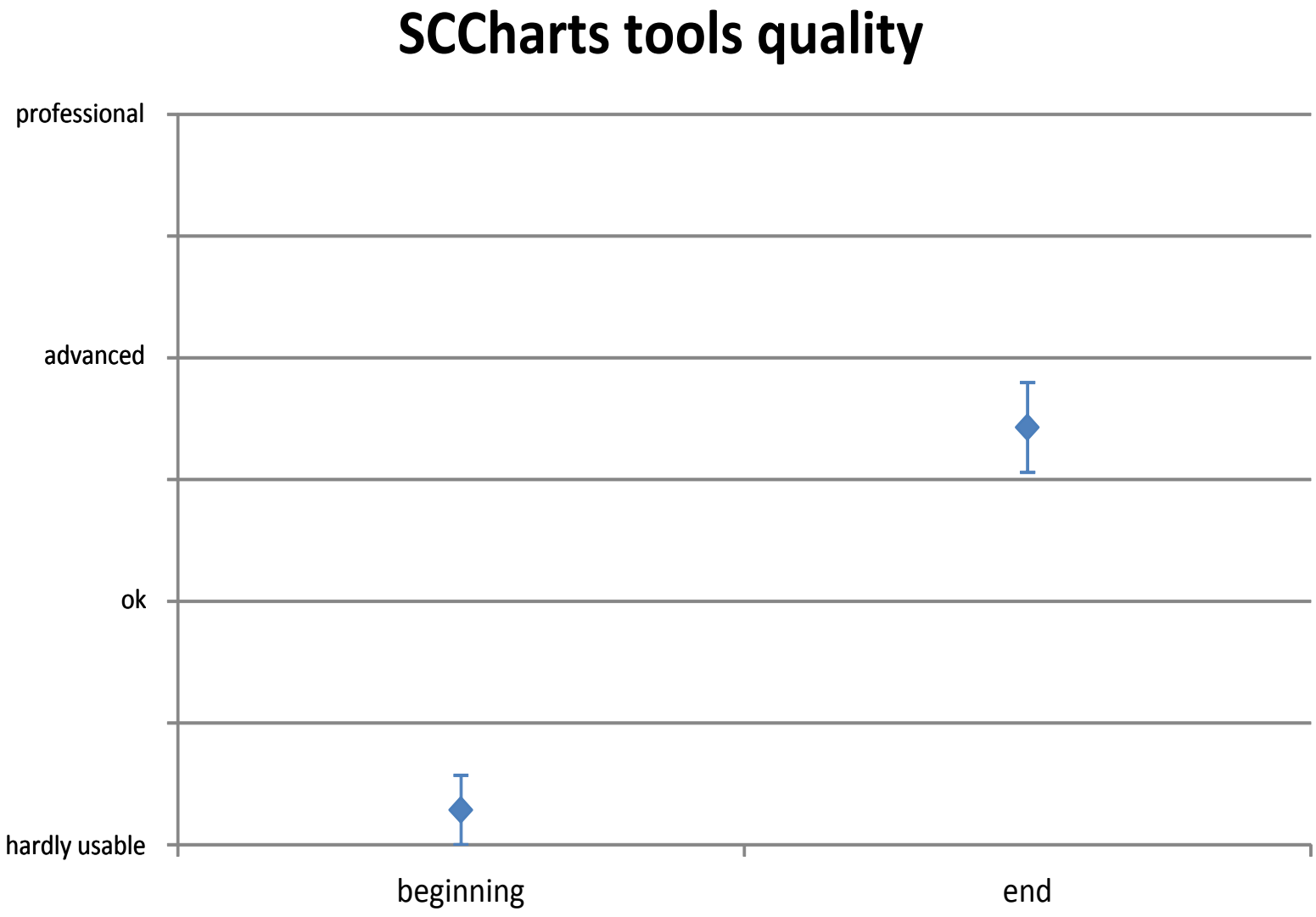
Which (extended) features did you miss?

- Generating multiple similar regions
  $\longrightarrow$ Map & For
- Broken features
  - complex final states x2
  - conditional termination
  - count delay
    $\longrightarrow$ Fix it!
- None x3
  $\longrightarrow$ Nice!

Overview

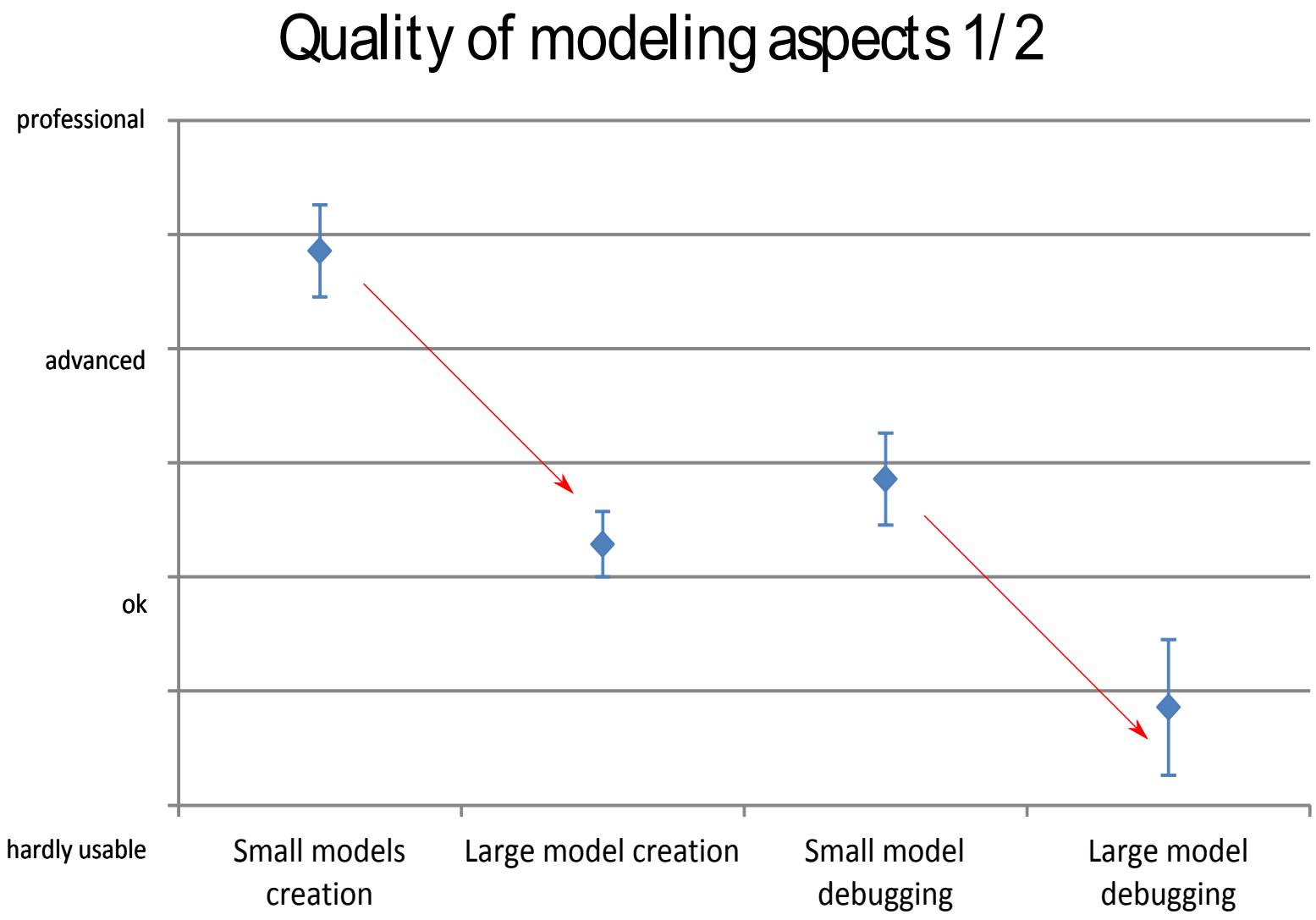
Tooling Evaluation

Your opinion about the overall quality of the SCCharts development tools...

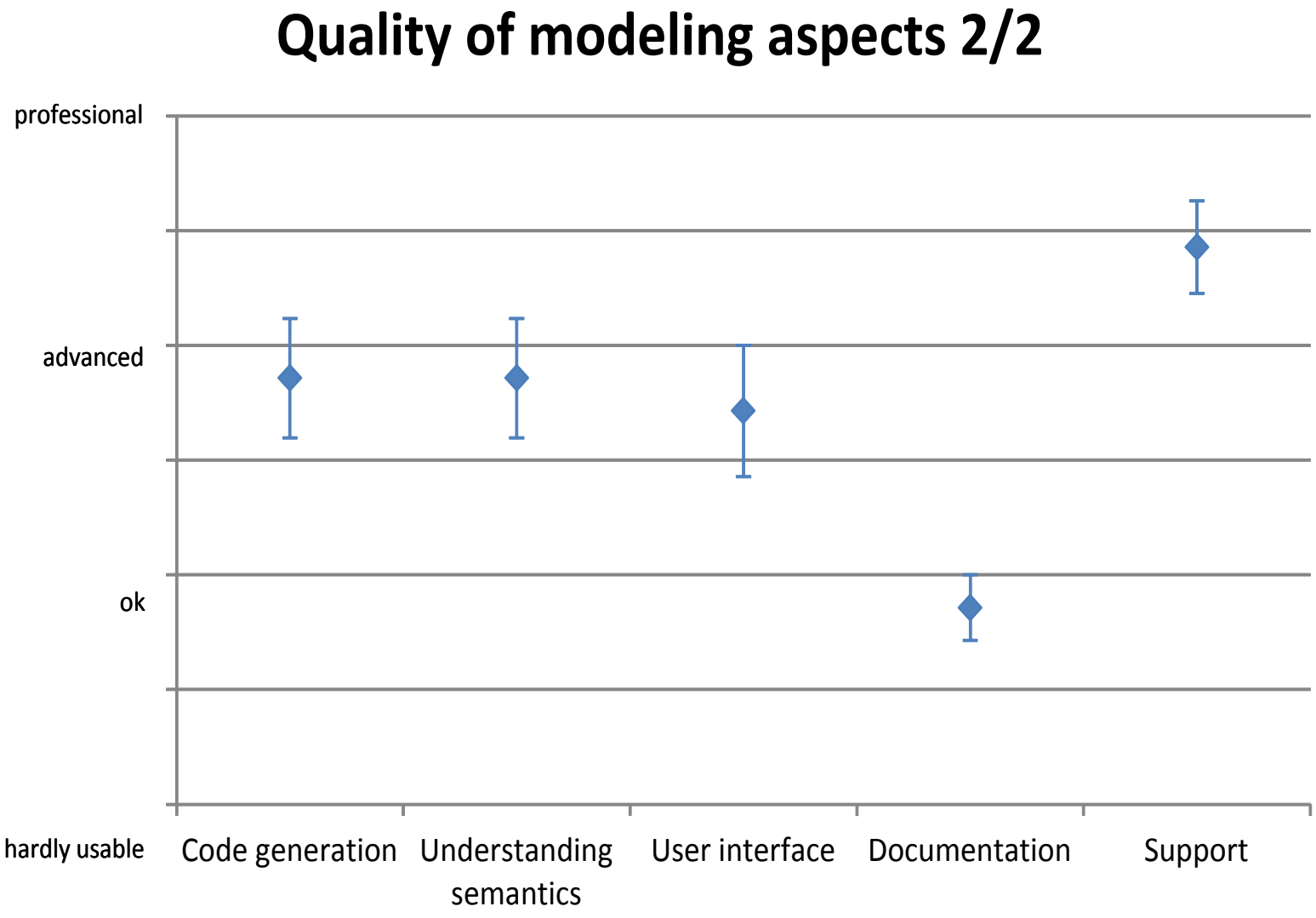


Tooling Evaluation

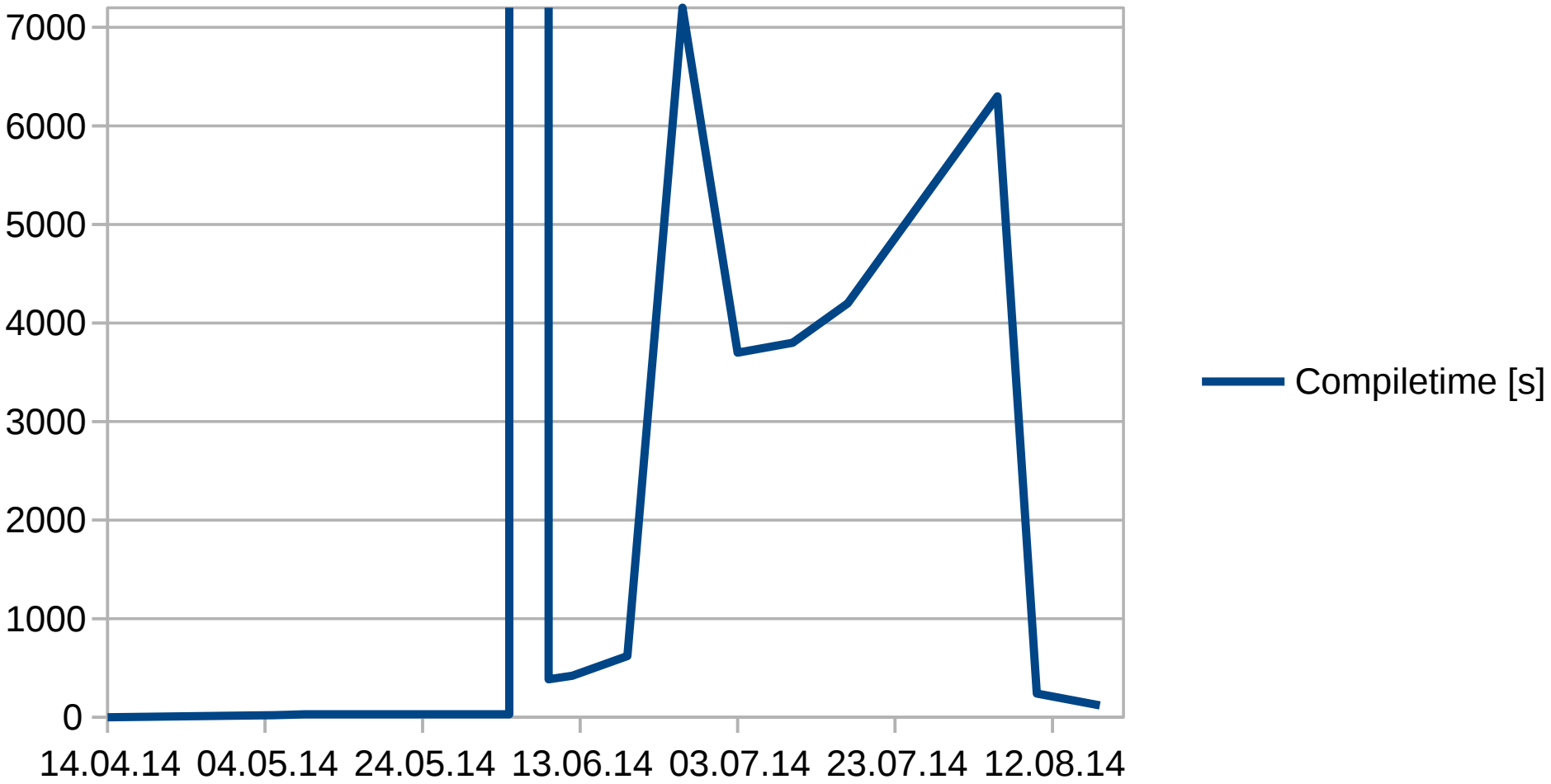
Tooling Evaluation



Tooling Evaluation



Tooling Evaluation - Compiler Performance



(taken from the final presentation of the railway project)

Tooling Evaluation - Compiler Performance

In a nutshell: Inefficient code

Tooling Evaluation - Compiler Performance

In a nutshell: Inefficient code

So far: focus on correctness and not efficiency. 

Tooling Evaluation - Compiler Performance

In a nutshell: Inefficient code

So far: focus on correctness and not efficiency. 

- Unnecessary content calls (*eAllContent, content, etc.*)

Tooling Evaluation - Compiler Performance

In a nutshell: Inefficient code

So far: focus on correctness and not efficiency. 

- Unnecessary content calls (*eAllContent, content, etc.*)
- Unnecessary list operations (*size, toList, etc.*)

Tooling Evaluation - Compiler Performance

In a nutshell: Inefficient code

So far: focus on correctness and not efficiency. 

- Unnecessary content calls (*eAllContent, content, etc.*)
- Unnecessary list operations (*size, toList, etc.*)
- Unwary use of ecore implementations (*ELists instead of Lists*)

Tooling Evaluation - Compiler Performance

In a nutshell: Inefficient code

So far: focus on correctness and not efficiency. 

- Unnecessary content calls (*eAllContent, content, etc.*)
- Unnecessary list operations (*size, toList, etc.*)
- Unwary use of ecore implementations (*ELists instead of Lists*)
- Over-complicated generic implementations (*generic* $\not>$ *specialized*)

Tooling Evaluation - Compiler Performance

In a nutshell: Inefficient code

So far: focus on correctness and not efficiency. 

- Unnecessary content calls (*eAllContent, content, etc.*)
- Unnecessary list operations (*size, toList, etc.*)
- Unwary use of ecore implementations (*ELists instead of Lists*)
- Over-complicated generic implementations (*generic* $\not>$ *specialized*)
- Missing caches, duplicated operations

Tooling Evaluation - Compiler Performance

In a nutshell: Inefficient code

So far: focus on correctness and not efficiency. 

- Unnecessary content calls (*eAllContent, content, etc.*)
- Unnecessary list operations (*size, toList, etc.*)
- Unwary use of ecore implementations (*ELists instead of Lists*)
- Over-complicated generic implementations (*generic* $\not>$ *specialized*)
- Missing caches, duplicated operations
- Unnecessary serializations in KiCo (extremely costly)

Tooling Evaluation - Compiler Performance

In a nutshell: Inefficient code

So far: focus on correctness and not efficiency. 

- Unnecessary content calls (*eAllContent, content, etc.*)
- Unnecessary list operations (*size, toList, etc.*)
- Unwary use of ecore implementations (*ELists instead of Lists*)
- Over-complicated generic implementations (*generic* $\not>$ *specialized*)
- Missing caches, duplicated operations
- Unnecessary serializations in KiCo (extremely costly)
- Often straight-forward approach (*e.g. dependency analysis*)

Tooling Evaluation - Compiler Performance

In a nutshell: Inefficient code

So far: focus on correctness and not efficiency. 

- Unnecessary content calls (*eAllContent, content, etc.*)
- Unnecessary list operations (*size, toList, etc.*)
- Unwary use of ecore implementations (*ELists instead of Lists*)
- Over-complicated generic implementations (*generic* $\not>$ *specialized*)
- Missing caches, duplicated operations
- Unnecessary serializations in KiCo (extremely costly)
- Often straight-forward approach (*e.g. dependency analysis*)
- Obfuscated bad code, mainly due to xtend extensions

```
val ctrlRegion = state.createRegion(GENERATED_PREFIX + "Ctrl").uniqueName
```

Tooling Evaluation - Compiler Performance

In a nutshell: Inefficient code

So far: focus on correctness and not efficiency. 

- Unnecessary content calls (*eAllContent, content, etc.*)
- Unnecessary list operations (*size, toList, etc.*)
- Unwary use of ecore implementations (*ELists instead of Lists*)
- Over-complicated generic implementations (*generic* $\not>$ *specialized*)
- Missing caches, duplicated operations
- Unnecessary serializations in KiCo (extremely costly)
- Often straight-forward approach (*e.g. dependency analysis*)
- Obfuscated bad code, mainly due to xtend extensions

```
val ctrlRegion = state.createRegion(GENERATED_PREFIX + "Ctrl").uniqueName
```

Bottom line: We need more reviews!

Tooling Evaluation - Compiler Performance

In a nutshell: Inefficient code

So far: focus on correctness and not efficiency. 

- Unnecessary content calls (*eAllContent, content, etc.*)
- Unnecessary list operations (*size, toList, etc.*)
- Unwary use of ecore implementations (*ELists instead of Lists*)
- Over-complicated generic implementations (*generic* $\not>$ *specialized*)
- Missing caches, duplicated operations
- Unnecessary serializations in KiCo (extremely costly)
- Often straight-forward approach (*e.g. dependency analysis*)
- Obfuscated bad code, mainly due to xtend extensions

```
val ctrlRegion = state.createRegion(GENERATED_PREFIX + "Ctrl").uniqueName
```

Bottom line: We need more reviews!

... and hopefully more sensitive towards efficiency.

Tooling Evaluation - Compiler Performance

In a nutshell: Inefficient code

So far: focus on correctness and not efficiency. 

- Unnecessary content calls (*eAllContent, content, etc.*)
- Unnecessary list operations (*size, toList, etc.*)
- Unwary use of ecore implementations (*ELists instead of Lists*)
- Over-complicated generic implementations (*generic* $\not>$ *specialized*)
- Missing caches, duplicated operations
- Unnecessary serializations in KiCo (extremely costly)
- Often straight-forward approach (*e.g. dependency analysis*)
- Obfuscated bad code, mainly due to xtend extensions

```
val ctrlRegion = state.createRegion(GENERATED_PREFIX + "Ctrl").uniqueName
```

Bottom line: We need more reviews!
... and hopefully more sensitive towards efficiency.

However, visualization of large models is another problem!

Tooling Evaluation

CONTRA remarks

Tooling Evaluation

CONTRA remarks

- Not all extended features are stable enough to use

Tooling Evaluation

CONTRA remarks

- Not all extended features are stable enough to use
- Still a lot of bugs/not working features

Tooling Evaluation

CONTRA remarks

- Not all extended features are stable enough to use
- Still a lot of bugs/not working features
- Debugging not schedulable models is a pain

Tooling Evaluation

CONTRA remarks

- Not all extended features are stable enough to use
- Still a lot of bugs/not working features
- Debugging not schedulable models is a pain
- Only one processor core is used

Tooling Evaluation

CONTRA remarks

- Not all extended features are stable enough to use
- Still a lot of bugs/not working features
- Debugging not schedulable models is a pain
- Only one processor core is used
- Big automata often confusing (visualization)

Tooling Evaluation

CONTRA remarks

- Not all extended features are stable enough to use
- Still a lot of bugs/not working features
- Debugging not schedulable models is a pain
- Only one processor core is used
- Big automata often confusing (visualization)
- Eclipse crashes

Tooling Evaluation

CONTRA remarks

- Not all extended features are stable enough to use
- Still a lot of bugs/not working features
- Debugging not schedulable models is a pain
- Only one processor core is used
- Big automata often confusing (visualization)
- Eclipse crashes
- SCCharts is a science project: language has it's roots in theory. Some features are theoretically very nice and interesting but in practice not really necessary.

Tooling Evaluation

PRO remarks

Tooling Evaluation

PRO remarks

- Good/Awesome support x2

Tooling Evaluation

PRO remarks

- Good/Awesome support x2
- Compiler finally fast enough x2

Tooling Evaluation

PRO remarks

- Good/Awesome support x2
- Compiler finally fast enough x2
- Neat and understandable for small automata

Tooling Evaluation

PRO remarks

- Good/Awesome support x2
- Compiler finally fast enough x2
- Neat and understandable for small automata
- SCCharts is a science project: quick bugfixing direct contact to developers

Overview

Outlook - Brainstorming

At the moment we're working on/thinking about...

Outlook - Brainstorming

At the moment we're working on/thinking about...

- Adaptive Zooming
  - needs fine tuning
  - however, is there a best fit?

Outlook - Brainstorming

At the moment we're working on/thinking about...

- Adaptive Zooming
  - needs fine tuning
  - however, is there a best fit?
- Lazy Loading for referenced SCCharts

Outlook - Brainstorming

At the moment we're working on/thinking about...

- Adaptive Zooming
  - needs fine tuning
  - however, is there a best fit?
- Lazy Loading for referenced SCCharts
- Transformation tracing
  - propagate compilation results to top level

Outlook - Brainstorming

At the moment we're working on/thinking about...

- Adaptive Zooming
  - needs fine tuning
  - however, is there a best fit?
- Lazy Loading for referenced SCCharts
- Transformation tracing
  - propagate compilation results to top level
- Further assessment of the project results
  - First large models
  - Survey results
  - Technical Report
  - Upcoming student theses

Outlook - Upcoming lectures

The next oportunities for SCCharts...

Outlook - Upcoming lectures

The next oportunities for SCCharts...

- Embedded System Design in WSem 14/15
  - Controlling the NXTs with SCCharts

Outlook - Upcoming lectures

The next oportunities for SCCharts...

- Embedded System Design in WSem 14/15
  - Controlling the NXTs with SCCharts
- Embedded System Project in SSem 15
  - Probably also an NXT project

The End

Thank you very much for your attention!