On the Pragmatics of Model-Based Design

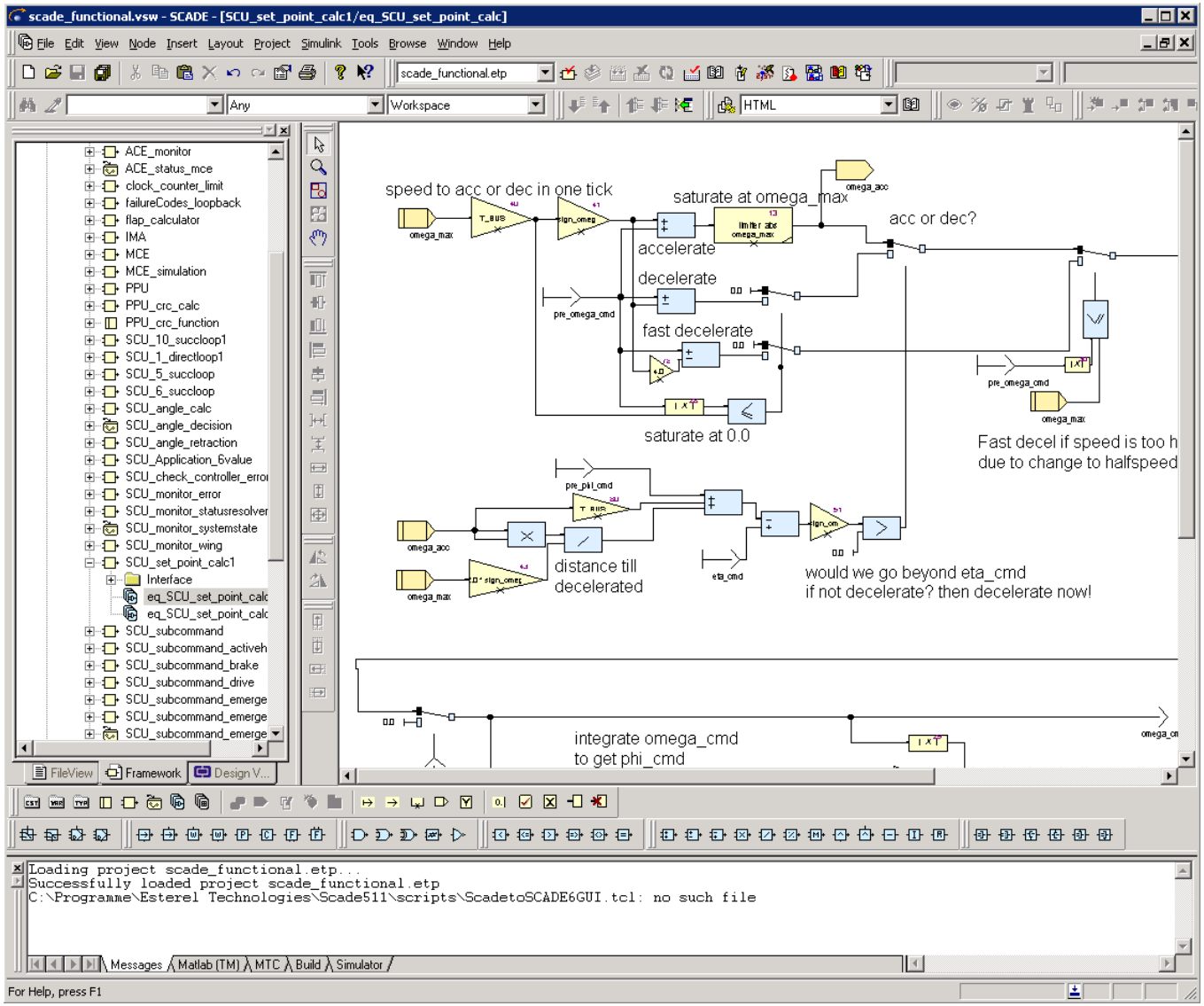
Position Statement

Reinhard von Hanxleden

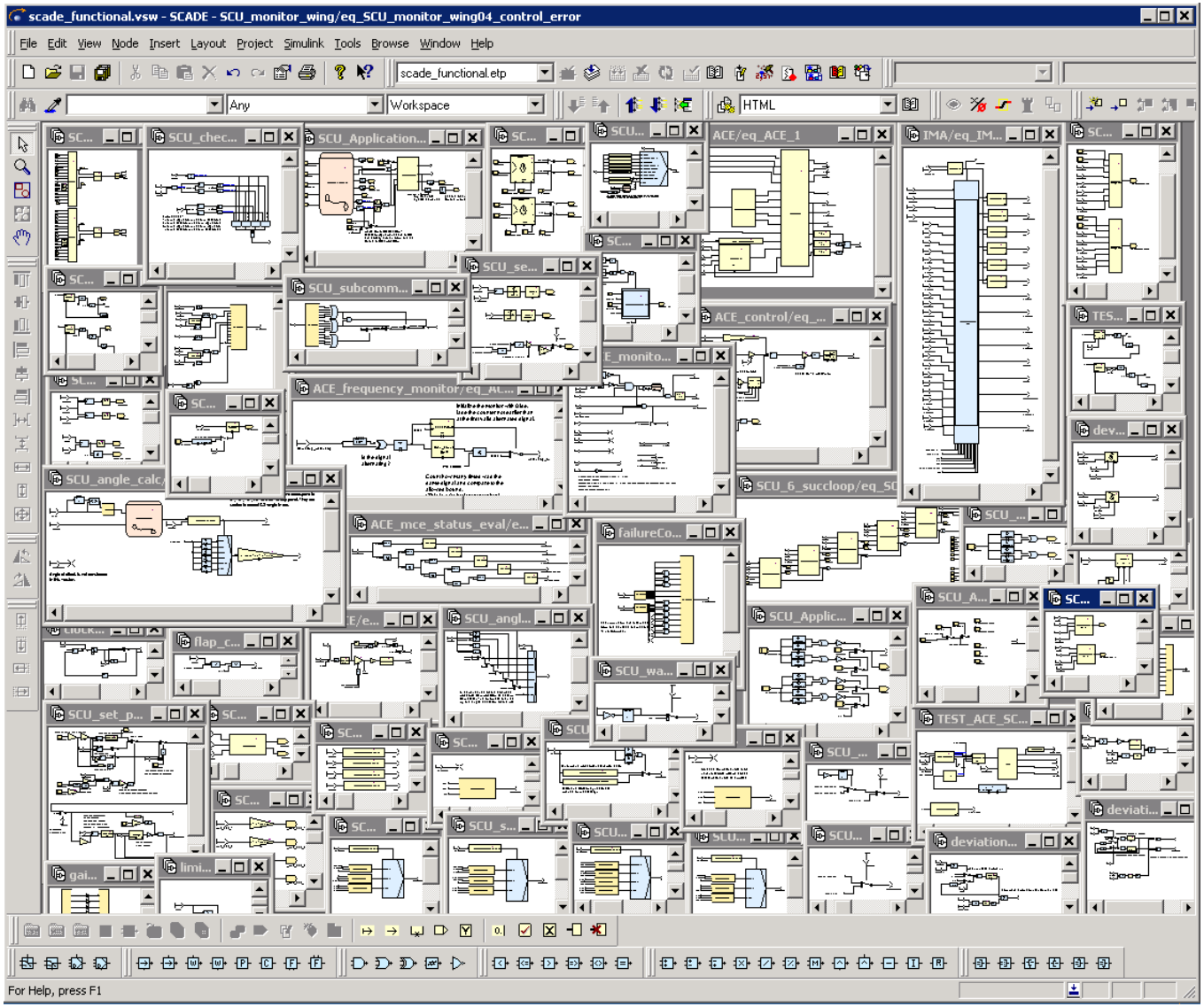
Real-Time Systems and Embedded Systems Group
Department of Computer Science
Christian-Albrechts-Universität zu Kiel, Germany
www.informatik.uni-kiel.de/rtsys

15th Monterey Workshop, Budapest, 25 September 2008

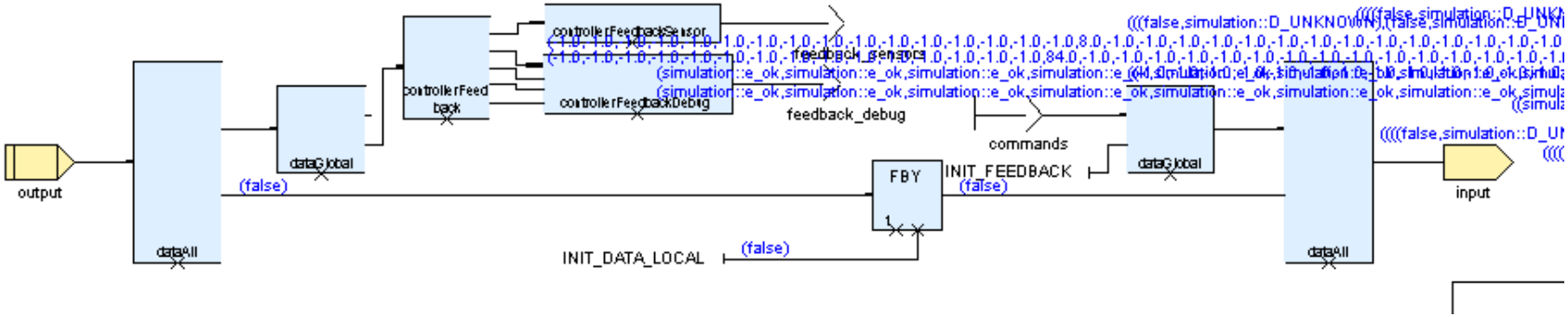
Institut für Informatik
Christian-Albrechts-Universität zu Kiel



Context missing

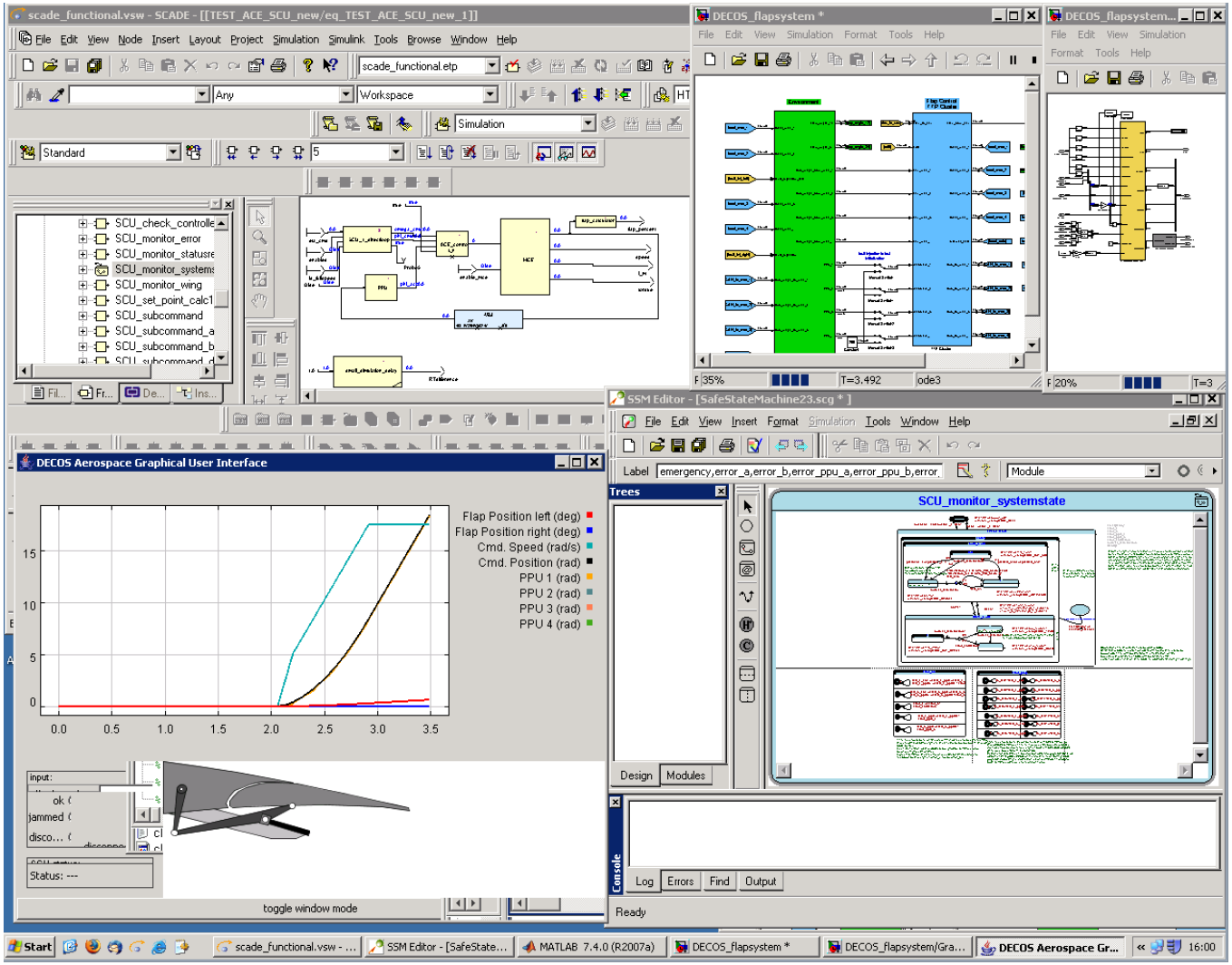


Quickly loose details



Data visualization difficult . . .

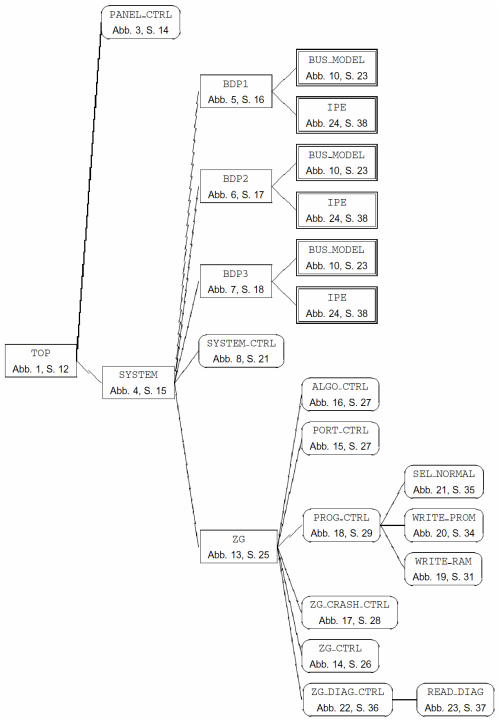
. . . to impossible

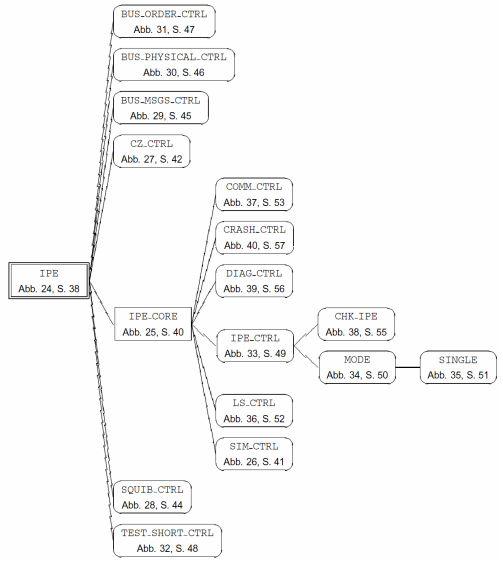


Screen real estate is tight!

Model-Based Design

Example of a complex system: Statemate model of airbag control





- Individually, 18 Activity Charts + 26 Statecharts
- After instantiation, $6 + 17 * 2 + 21 = 61$ Activity Charts + $12 + 17 * 15 + 21 = 288$ Statecharts!

Overview

Position Statement

My Position: Pragmatics of modeling languages deserves more attention than it has received so far

- Specifically: practical issues of how to create, maintain, browse and visualize graphical models have been neglected in the past.
- This
  1. largely offsets the inherent advantages of visual languages,
  2. unduly limits designers' productivity, and
  3. makes it difficult to design complex systems.

Pragmatics of Model-Based Design

Pragmatics: relation of signs to their users

$+$

Syntax: relations between signs

$+$

Semantics: relations between signs and the things they refer to

$=$

Semiotics: how meaning is constructed and understood

(Charles Morris, *Foundation of the Theory of Signs*, 1938)

Pragmatics of Model-Based Design

Pragmatics usually concentrates on practical aspects of how constructs and features of a language may be used to achieve various objectives (*e. g.*, when to use an assignment).

Here, will focus on the mechanics of handling a language (editing, maintaining, inspecting).

Pragmatics of modeling languages $=_{def}$
practical aspects of handling a model in a model-based design flow

The Big Picture

The vision:

- Provide flexible, alternative views of system under development (SUD)
- Free the designer from tedious model editing tasks

The approach:

- Get inspiration from successful textual paradigms and tools
- Combine best of graphical and textual worlds
- Use Model-View-Controller pattern

The key enabler:

- Automatic, flexible synthesis of graphical models

Overview

The Model-View-Controller (MVC) Paradigm

Models Models represent knowledge. A model could be a single object (rather uninteresting), or it could be some structure of objects.

Views A view is a (visual) representation of its model. It would ordinarily highlight certain attributes of the model and suppress others. It is thus acting as a *presentation filter*.

Controllers A controller is the link between a user and the system. It provides the user with input by arranging for relevant views to present themselves in appropriate places on the screen.

(Trygve Reenskaug, *Models – Views – Controllers*, Xerox PARC technical note, 1979)

MVC for Model-Based Design

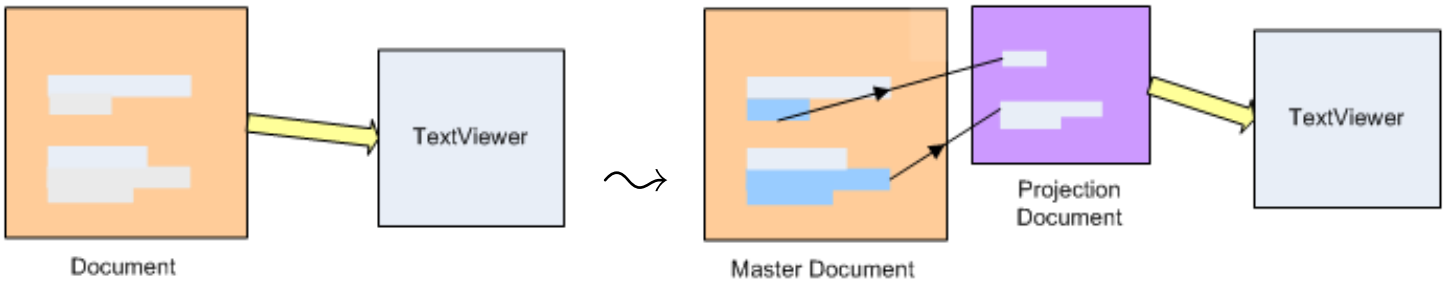
- Today, MVC is a well-established SW engineering paradigm (*MVC for tool developers*)
- Eg, it may typically be employed for the design of a modeling tool, as guiding principle when developing the tool
  - Model: Current state of tool, data structures for file handling, etc.
  - View: GUI of tool
  - Controller: Tool driver
- Proposal: Employ MVC also for the design of an embedded system model, as guiding principle in a model-based design process (*MVC for tool users*)!
  - Model: Model of the System-Under-Development (SUD)
  - View: Visualization of SUD during editing, simulation, etc.
  - Controller: The modeling tool

MVC in Model Editing

Current state: Graphical WYSIWYG editors

Alternatives:

- Structure-based editors
- Text-based editors
- Layout and meta-layout
- Folding editors



Prashant Deva.
Folding in Eclipse Text Editors.
http://www.eclipse.org/articles/Article-Folding-in-Eclipse-Text-Editors/folding.html

MVC in Model Simulation

Current state: Coloring of static view

Alternatives:

- Dynamic semantic focus and context representation
- Promote visualization (layout) information and simulation control to first-class citizen—*e. g.*, "if in(error) then show(diagnostics)"
- Model annotations, visualization/simulation scripting (cf. OCL)

MVC in Model Analysis & Documentation

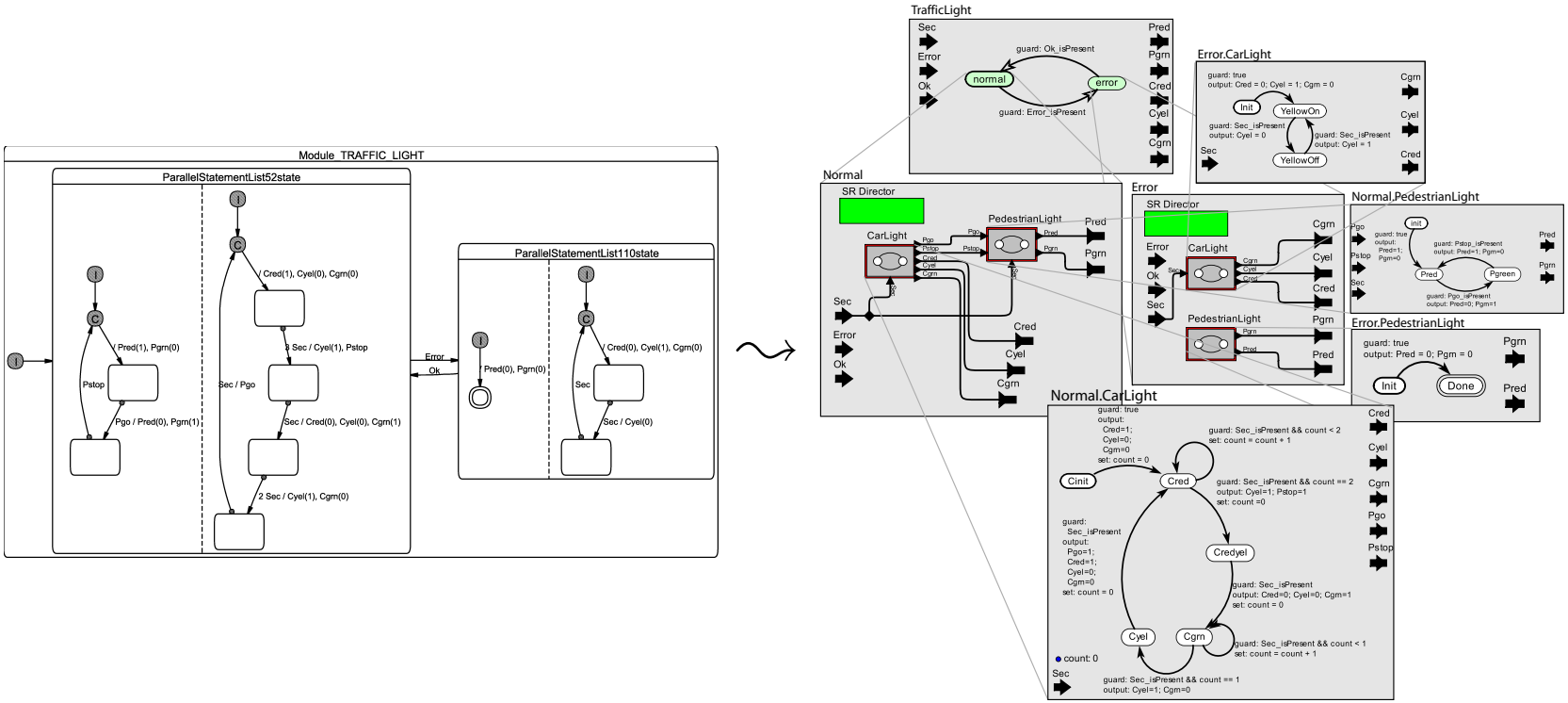
Current state: Inspection of static view

Alternatives:

- Literate modeling
- Multi-view modeling: Functional model vs. deployment model vs. verification model

Multi-View Modeling

Example: Traffic-Light Controller



C. Brooks, C. P. Cheng, T. H. Feng and E. A. Lee, R. von Hanxleden.
Model Engineering using Multimodeling.
Proceedings of the 1st International Workshop on Model Co-Evolution and Consistency Management (MCCM'08), a workshop at MODELS'08, September 2008.

Conclusion & Outlook

- Current practice limits productivity in model-based design
- Synthesis of (views of) graphical models is a bottle neck
- Automatic layout seems feasible
- MVC paradigm considered helpful
- First ideas realized in Kiel Integrated Environment for Layout
- Next steps: KIEL for the Eclipse RichClientPlatform (KIELER)
- `http://www.informatik.uni-kiel.de/rtsys/kieler/`

thanks!

questions or comments?