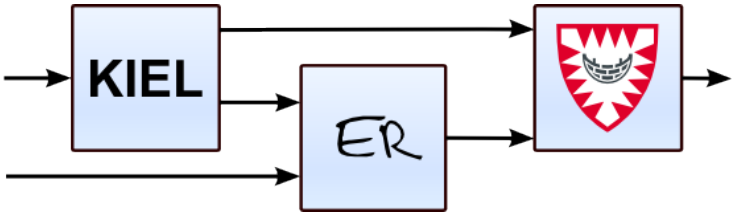
Multi-View Modeling and Pragmatics in 2020

Position Paper on Designing Complex Cyber-Physical Systems

Reinhard von Hanxleden (U Kiel), Edward A. Lee (UC Berkeley),
Christian Motika (U Kiel), Hauke Fuhrmann (Funkwerk)

17th Monterey Workshop, March 19–21, Oxford, UK

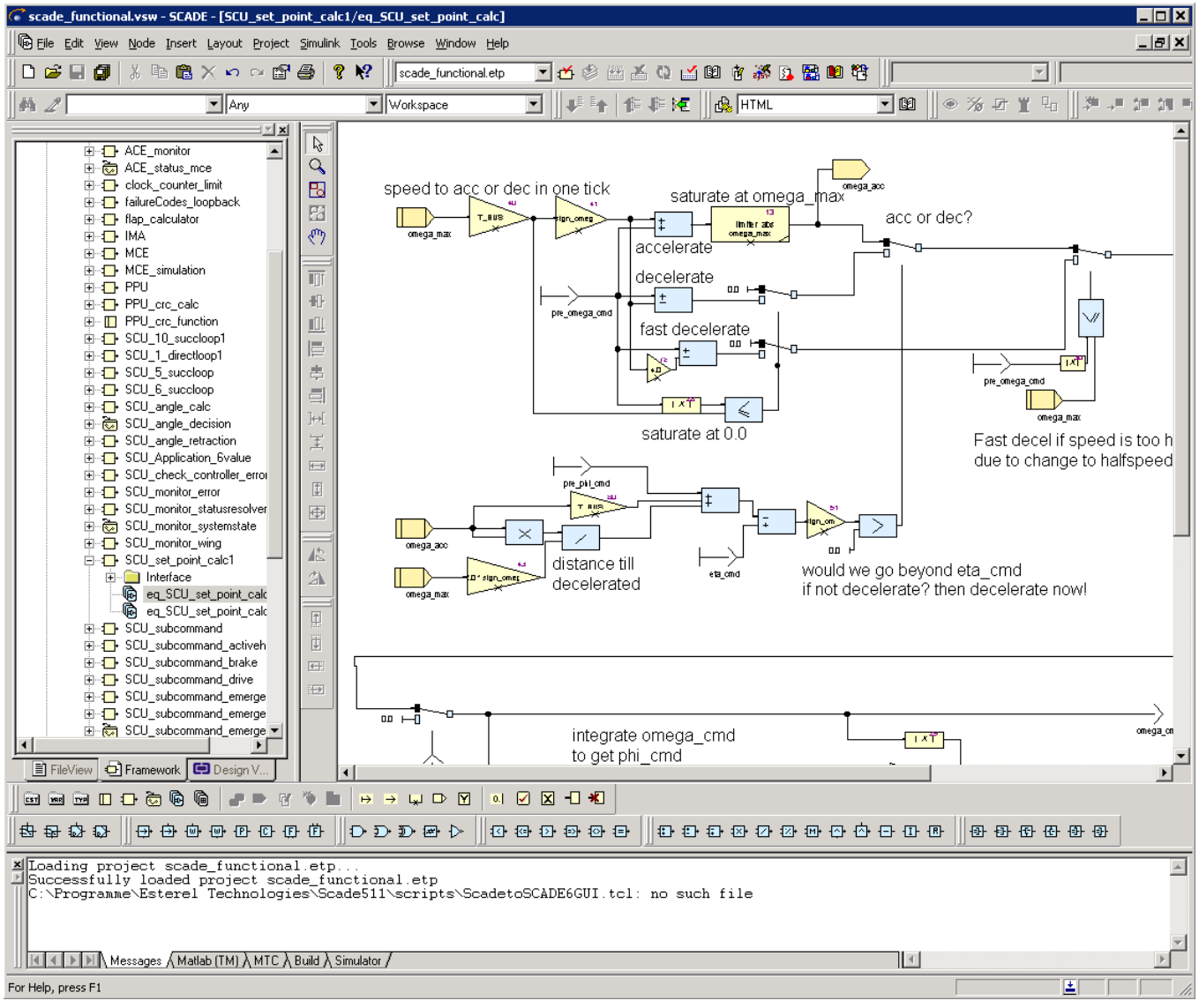


Kiel **I**ntegrated **E**nvironment for **L**ayout

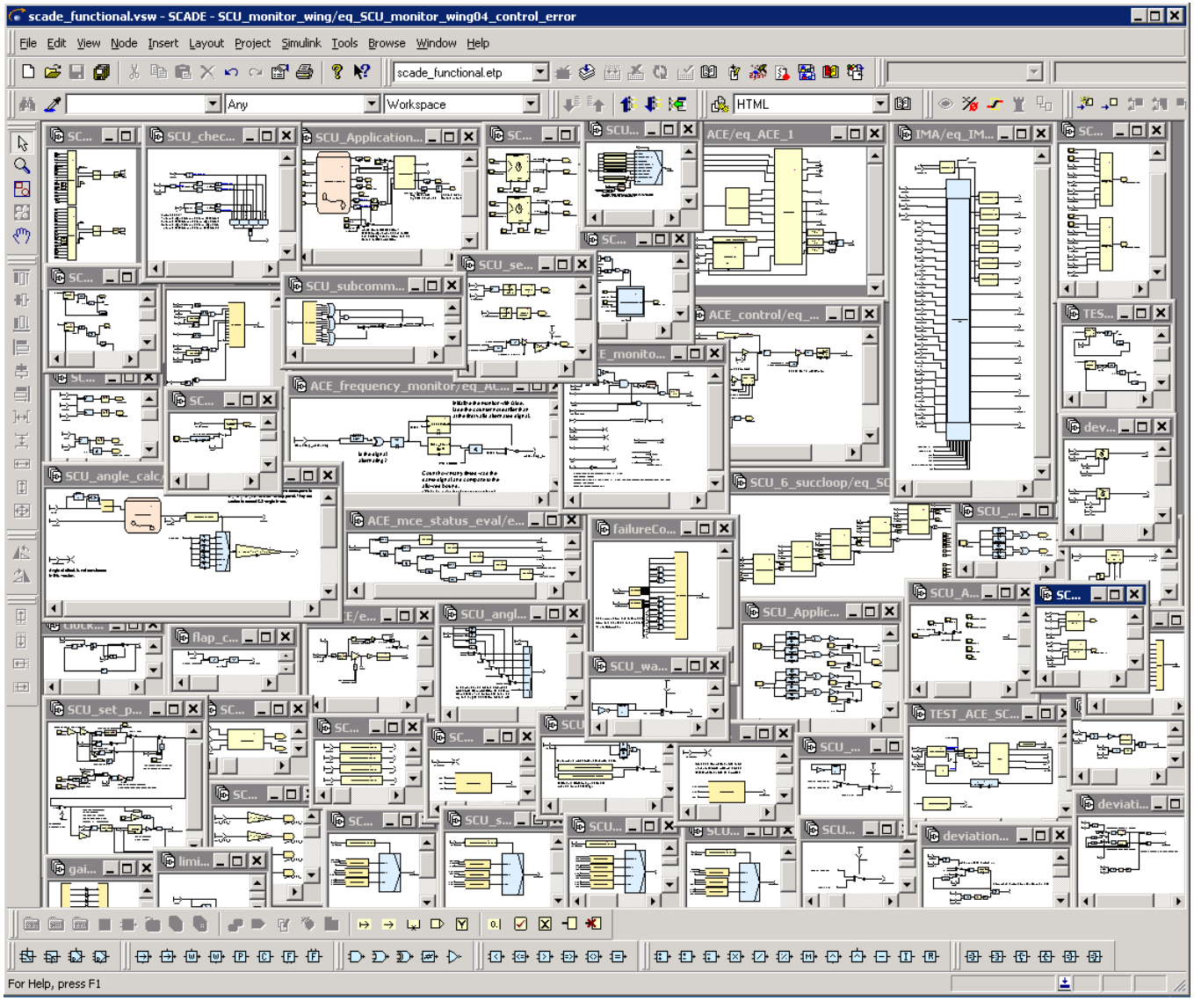
Eclipse Rich Client

Designing Complex Cyber-Physical Systems:

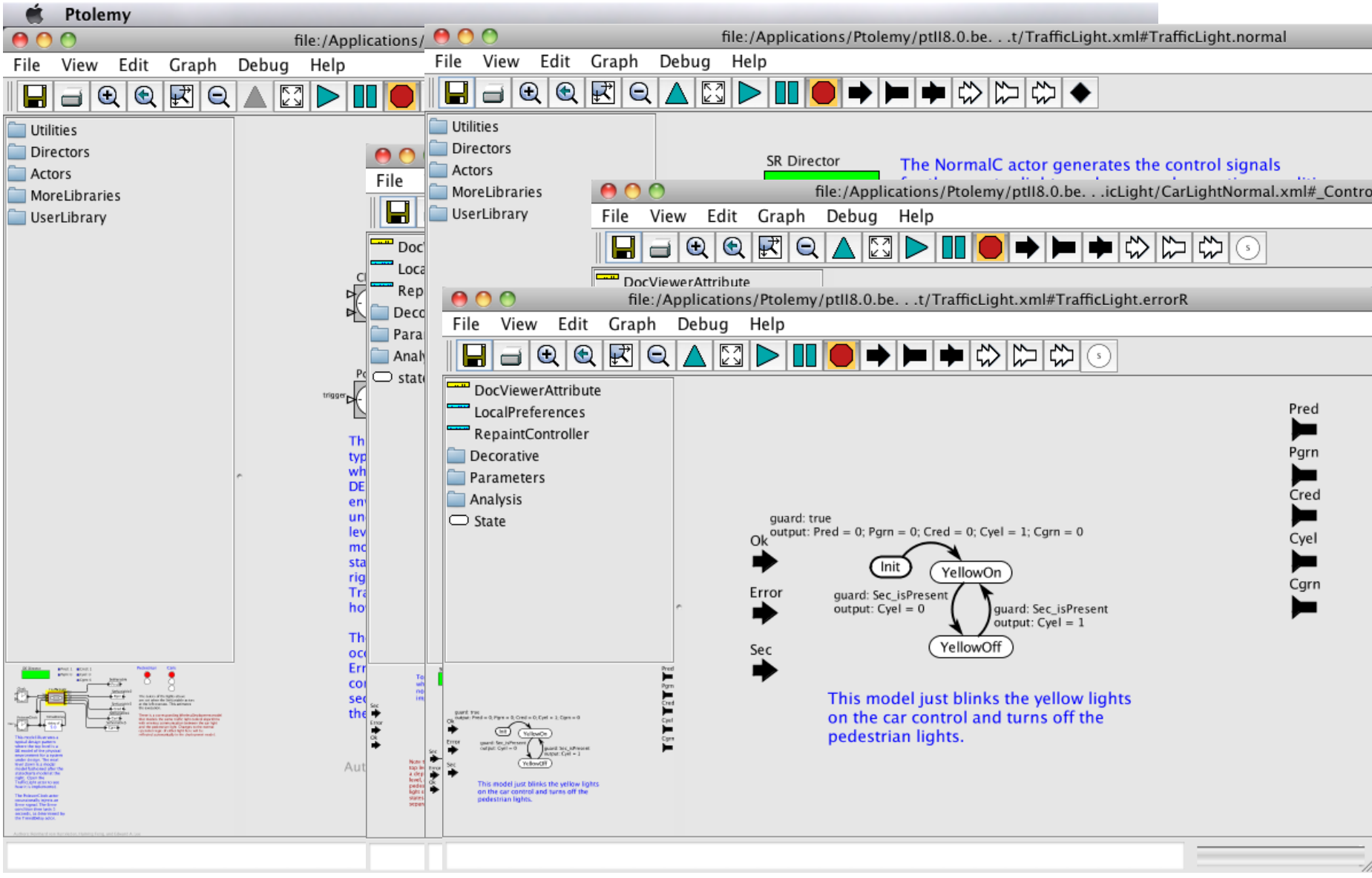
(Some) Issues



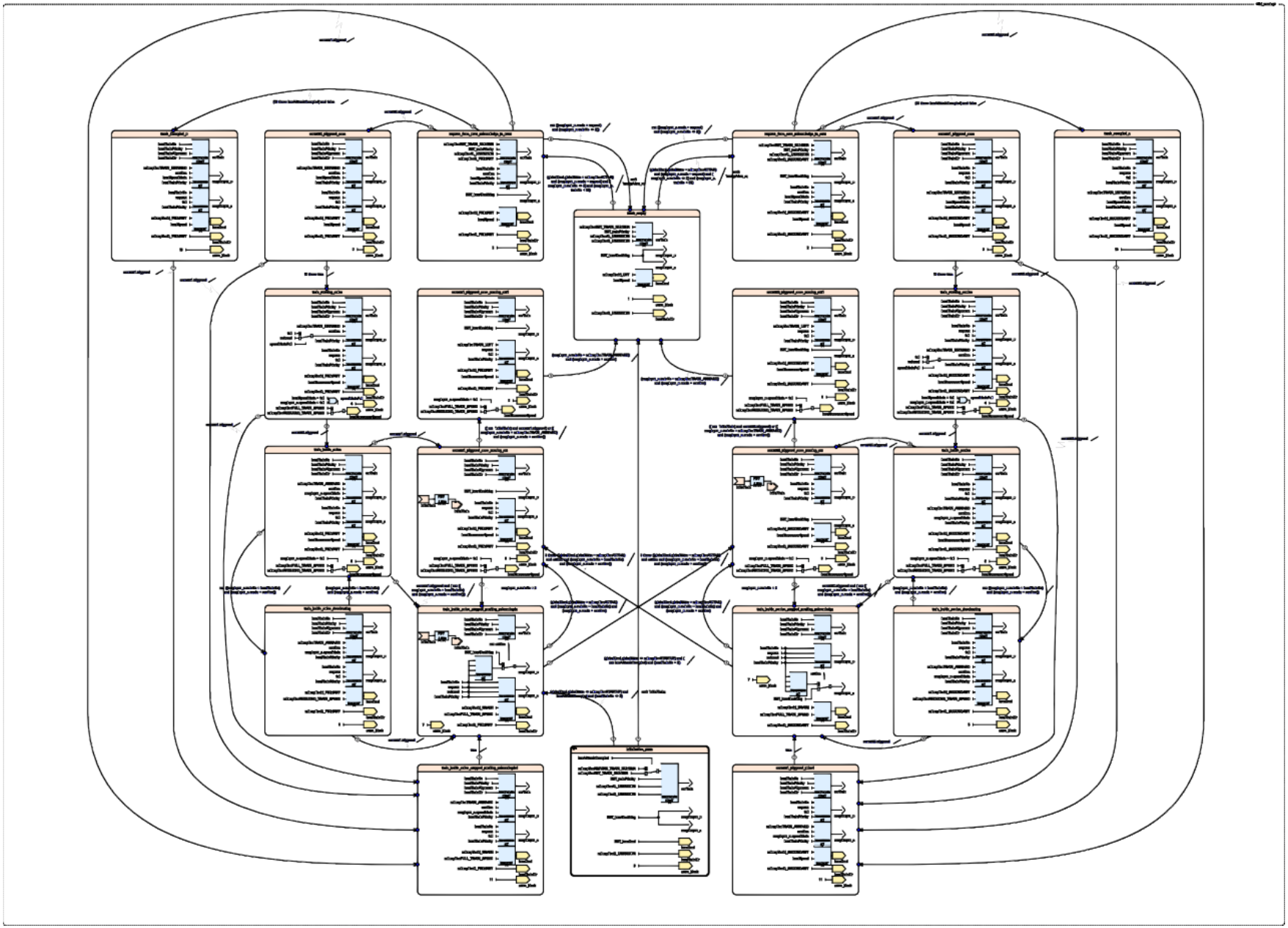
Context missing

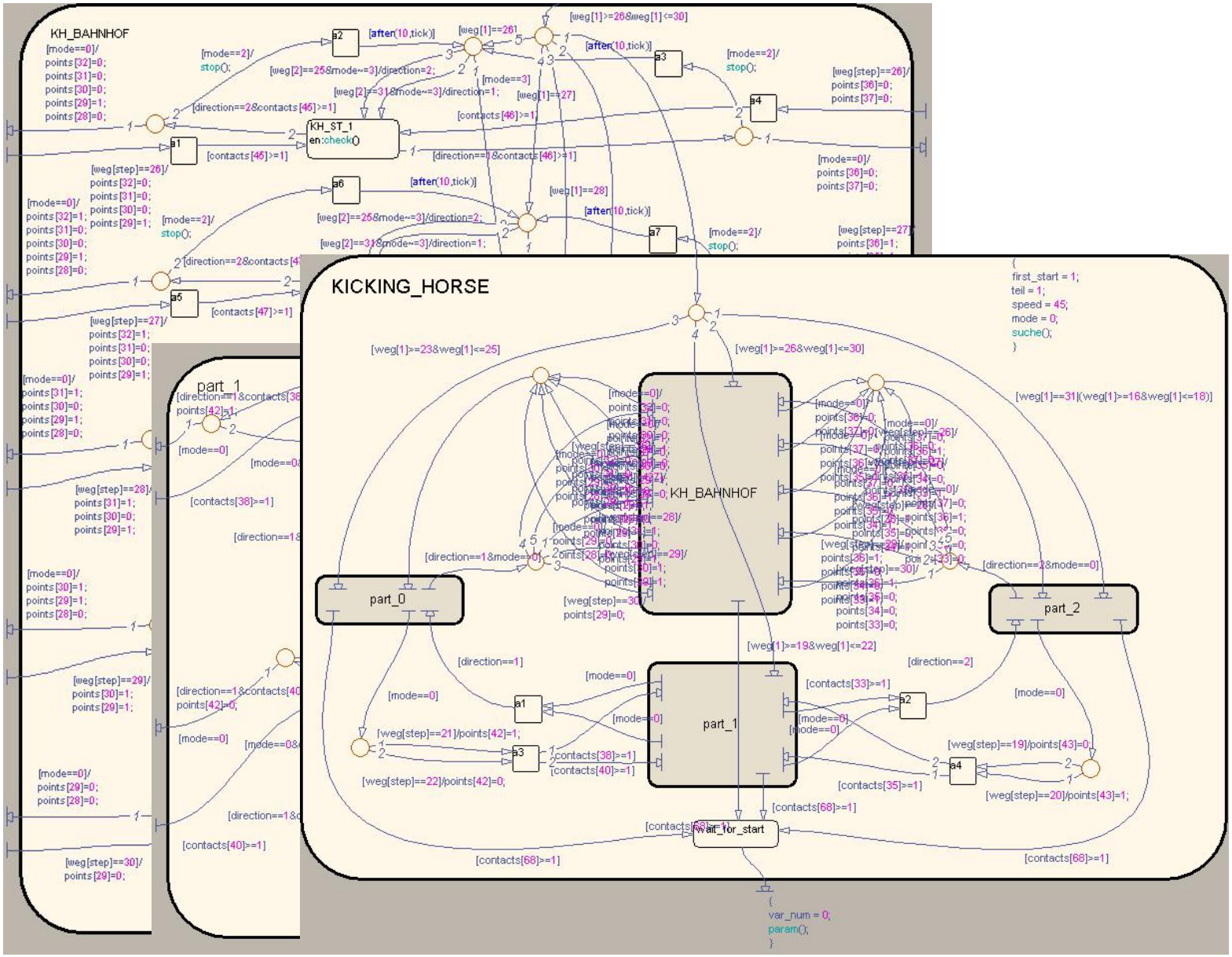


Quickly loose details

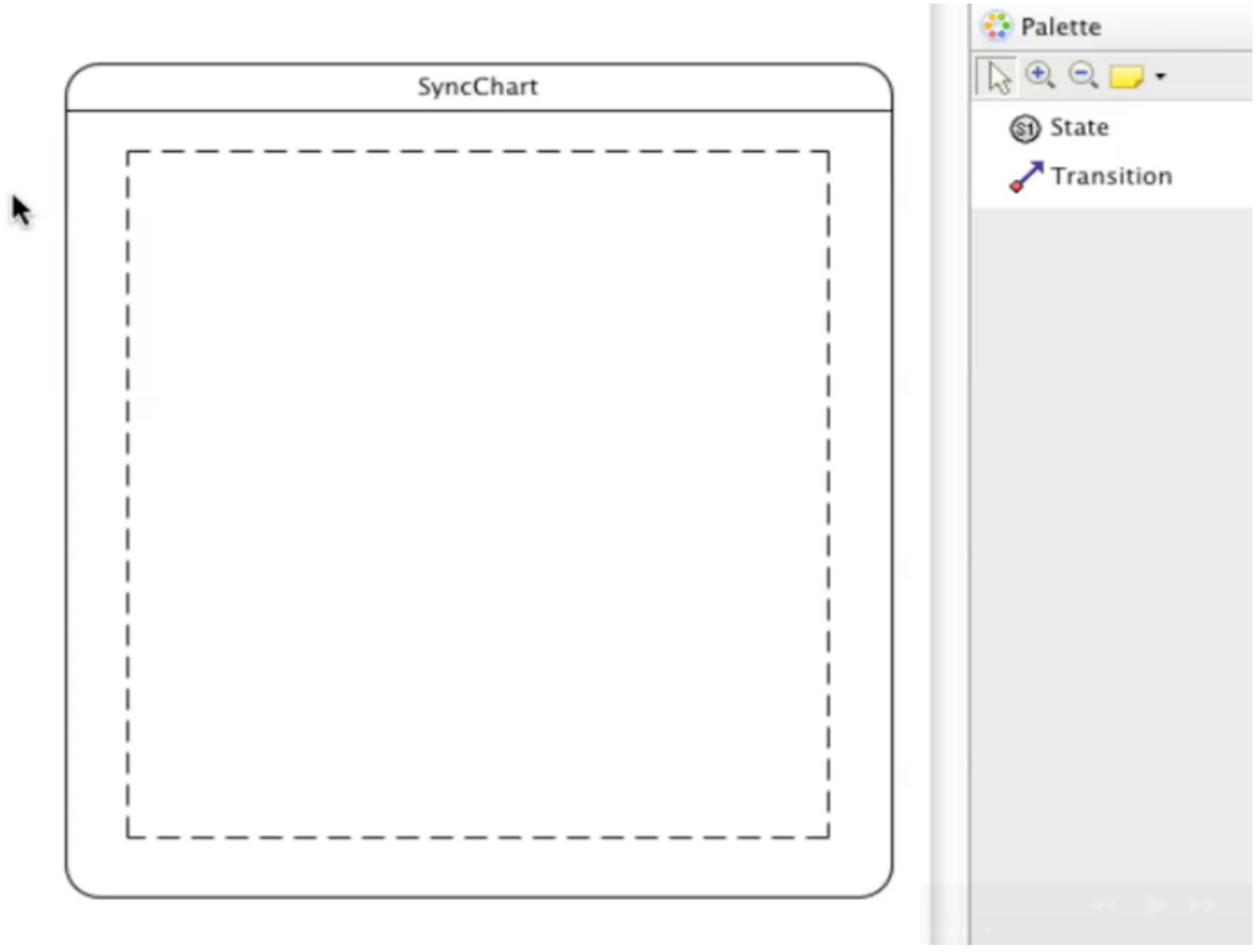


Model hierarchy translates into a cluttered screen





. . . or too little!



Editing can be sloooow . . .

Position: Separating *Model* and *View* crucial for managing complexity

Overview

Models, Views, Controllers

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

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

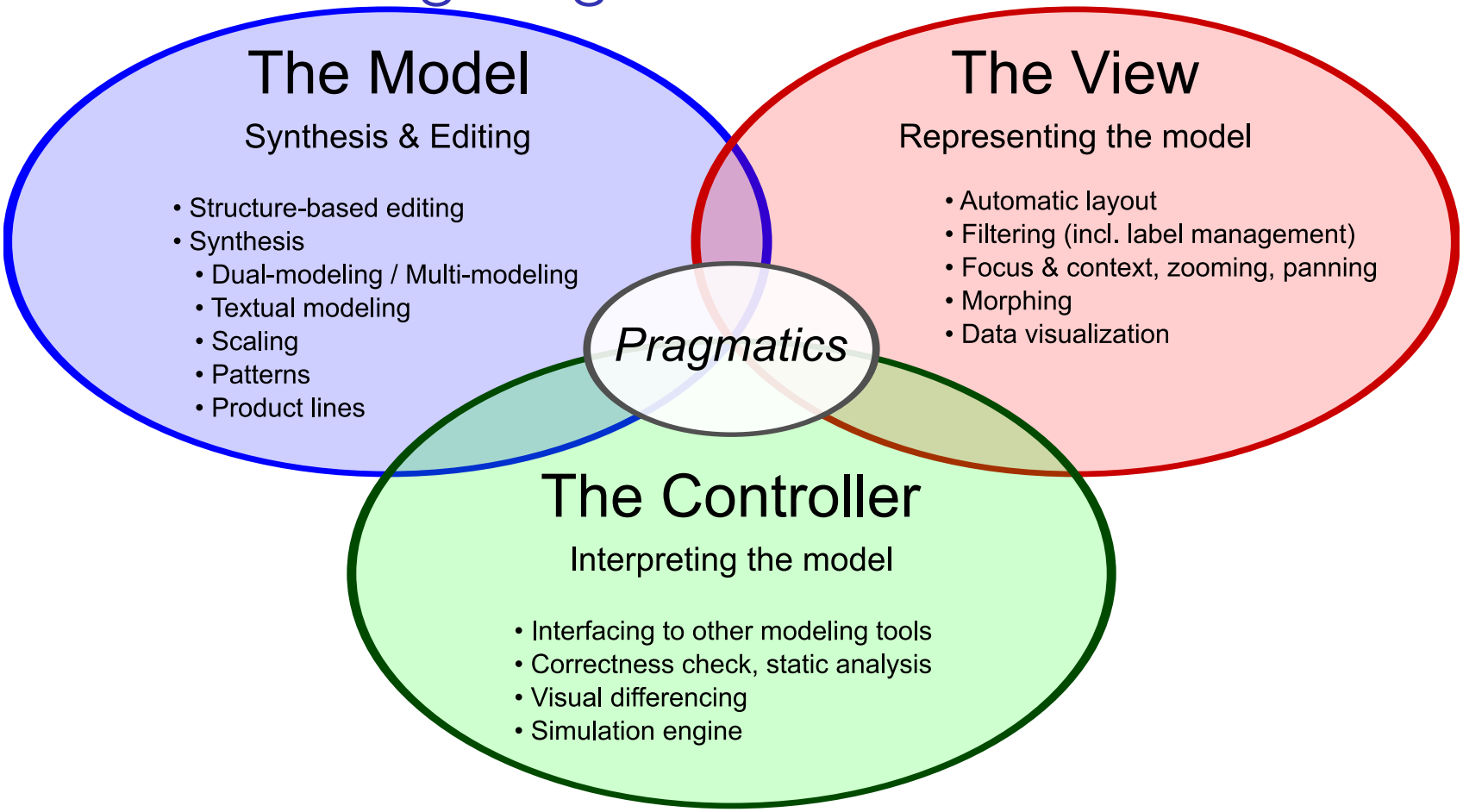
$=$

Semiotics: how meaning is constructed and understood

Charles Morris.
Foundation of the Theory of Signs.
University of Chicago Press, 1938

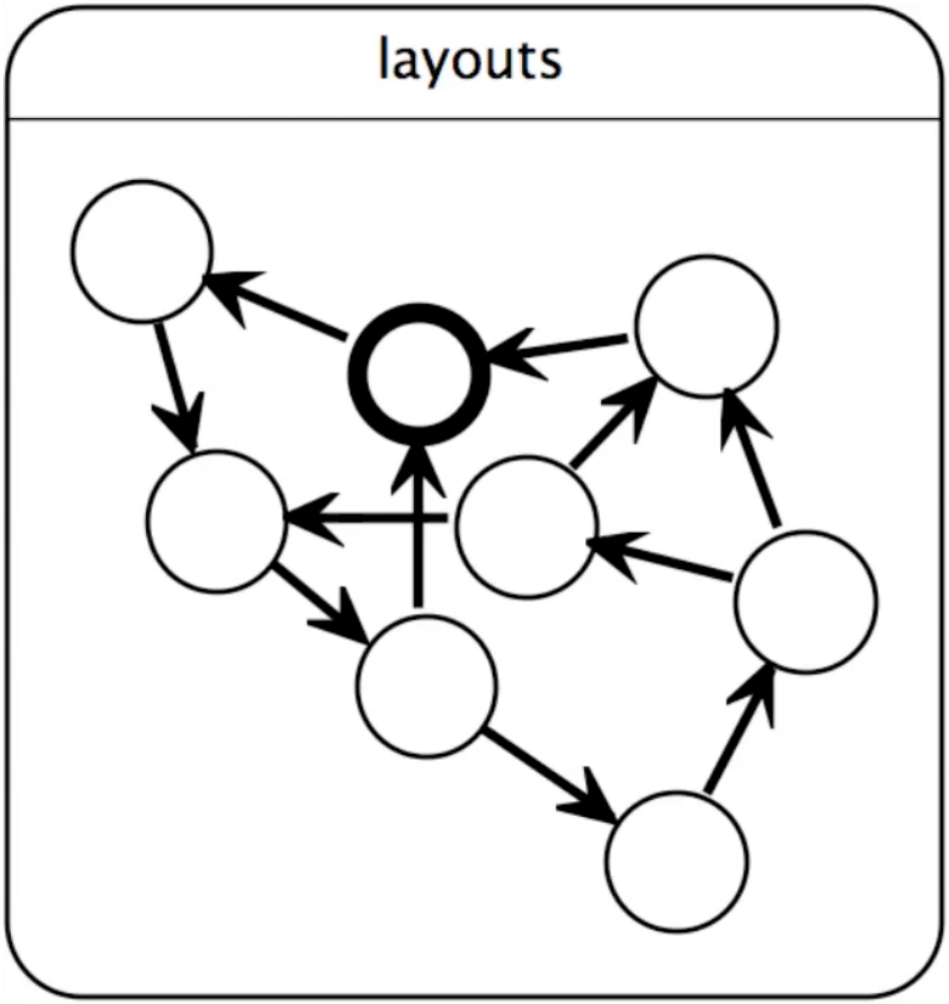
Pragmatics of modeling languages $=_{def}$
practical aspects of handling a model in a model-based design flow

MVC and Modeling Pragmatics



Hauke Fuhrmann and Reinhard von Hanxleden.
On the Pragmatics of Model-Based Design.
15th Monterey Workshop 2008, Budapest, Hungary, September 24–26, 2008

Key to Separate Models and Views: Automatic Layout



Overview

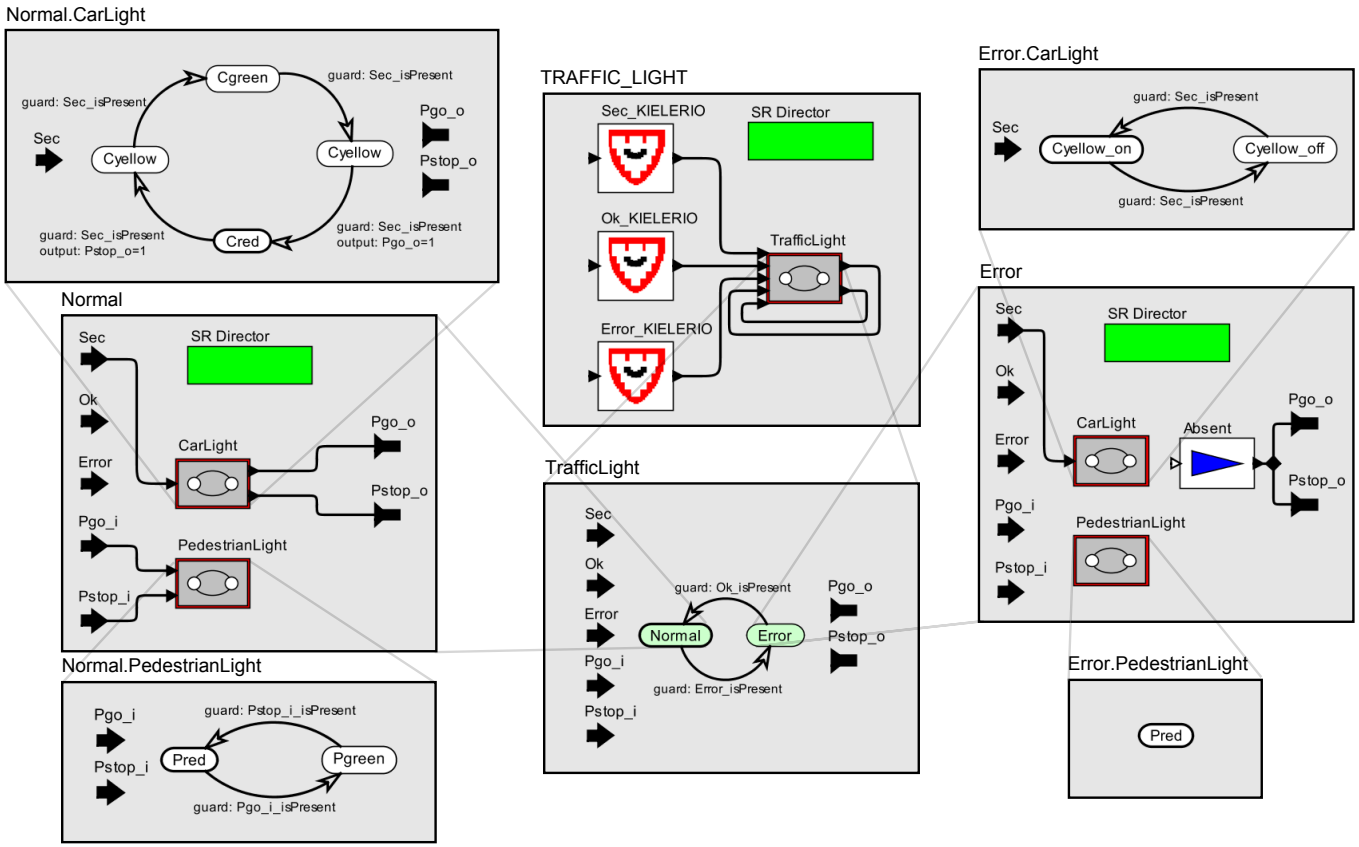
Trend 1: Agile, Domain-Specific Development Processes

- Monolithic one-way methods $\longrightarrow$ agile, iterative processes
- Big, one-size fits all frameworks and languages $\longrightarrow$ DSLs

2020 Vision:

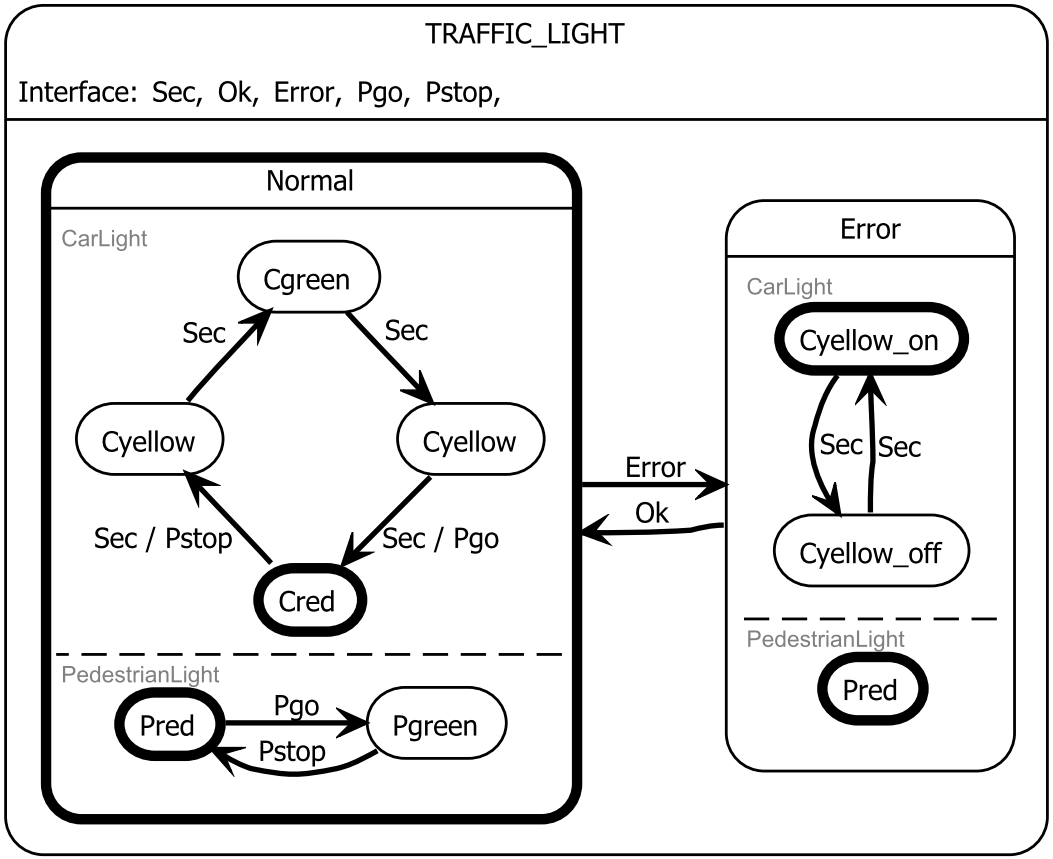
- Usage-specific views
- Usage-specific languages

Example: Traffic-Light Controller



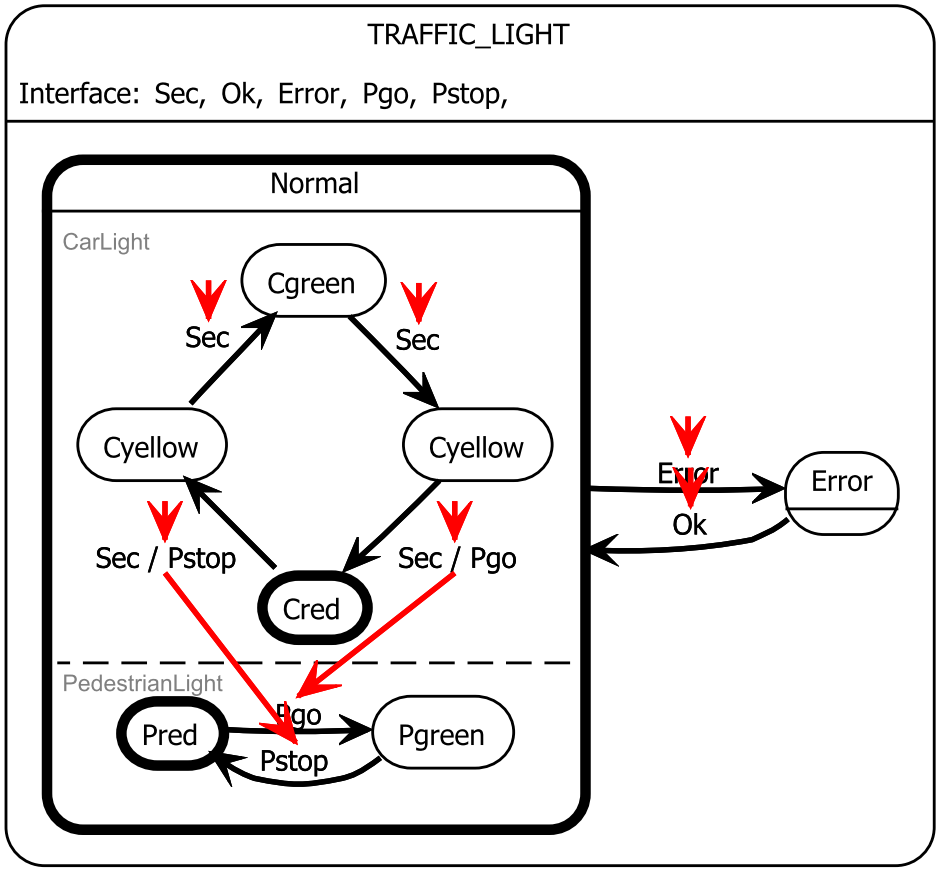
Structural view (hierarchical data-flow + automata)

Example: Traffic-Light Controller



Behavioral View (SyncChart)

Example: Traffic-Light Controller



Hybrid view (SyncChart + dual modeling + focus&context filtering)

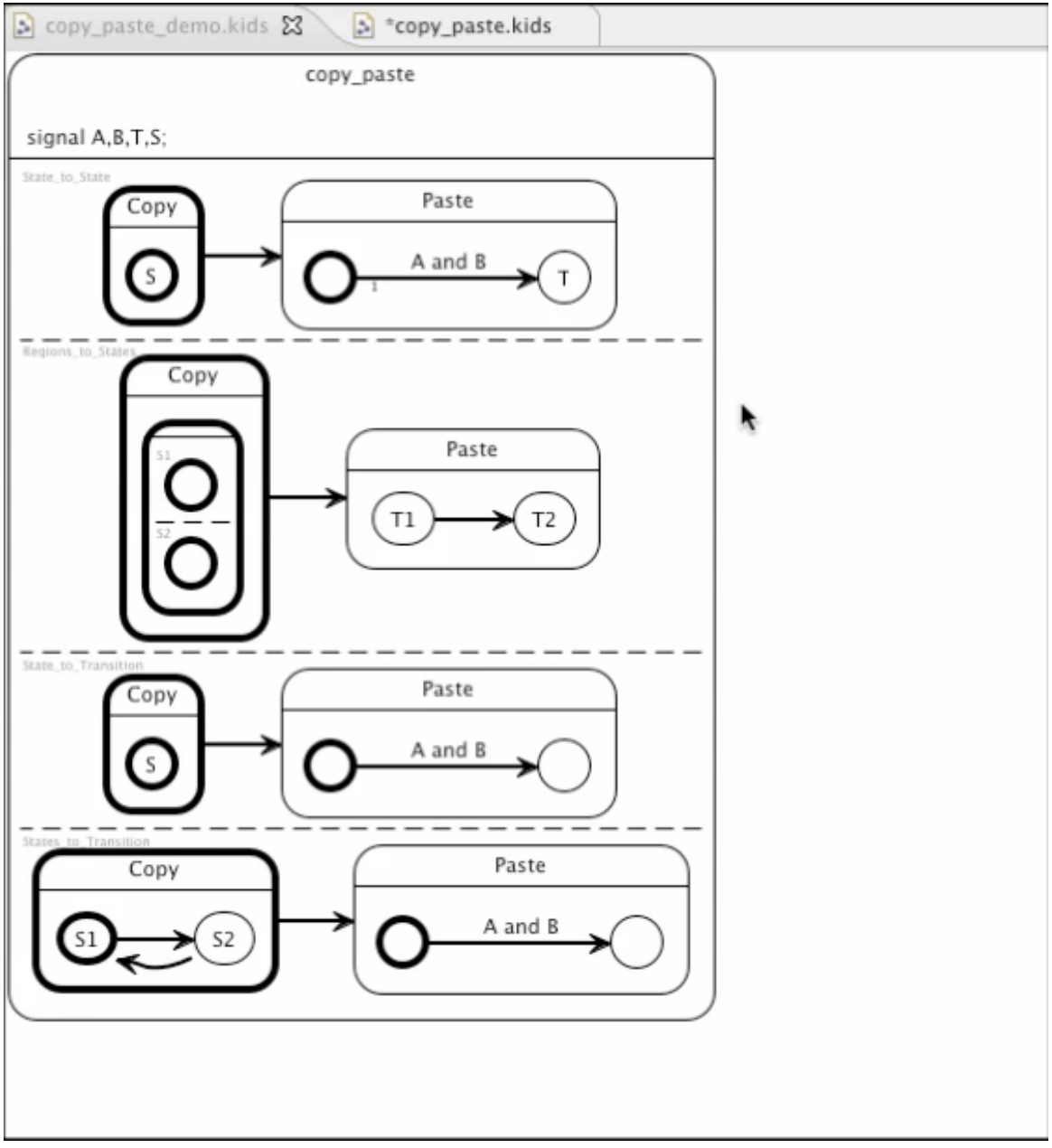
Trend 2: Novel Input Devices

- Post-PC devices
- Technological enablers for intuitive interaction paradigms

2020 Vision:

- Touch-based editing and browsing
- Move from *location-based editing* to *object-based editing*

Example: Advanced Copy & Paste



Trend 3: The Move to the Cloud

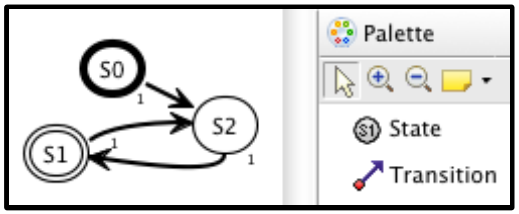
- No lengthy installation procedures
- Always current tool version

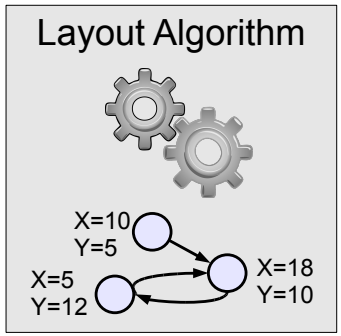
2020 Vision:

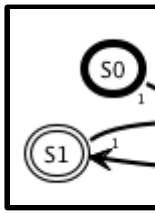
- Actor-oriented, cloud-based modeling tools

Example: Automatic Layout as Remote Service

Diagram Editor View







- Eclipse GMF
- Graphiti (ongoing)
- Ptolemy
- ...

- GraphViz (Dot,Neato,FDP,Twopi,Circo,Radial)
- Open Graph Drawing Framework (OGDF) (Class Diagram, Layer-Based, Force Directed, Orthogonal, Planarization,...)
- Zest (GEF)
- Own Implementations (Ports, Layer-Based, Planarization,...)
- ...

Overview

Conclusion & Outlook

- Separating models and views is key to handling complexity
- Trends:
  1. Agile processes $\Longrightarrow$ usage-specific languages/views
  2. Novel input devices $\Longrightarrow$ object-based editing
  3. The cloud $\Longrightarrow$ (eg.) layout as a service
- KIELER is laboratory for exploring pragmatics of model-based design (EPL)
- `http://www.informatik.uni-kiel.de/rtsys/kieler/`

thanks!

questions or comments?

Appendix

Pragmatics

- Pragmatics – Syntax – Semantics – Semiotics
- Pragmatics of Model-Based Design
- An Experiment

KIELER

- Building on Automatic Layout: View Management
- KIELER Semiotics
- Putting KIELER to Work

Pragmatics of Model-Based Design

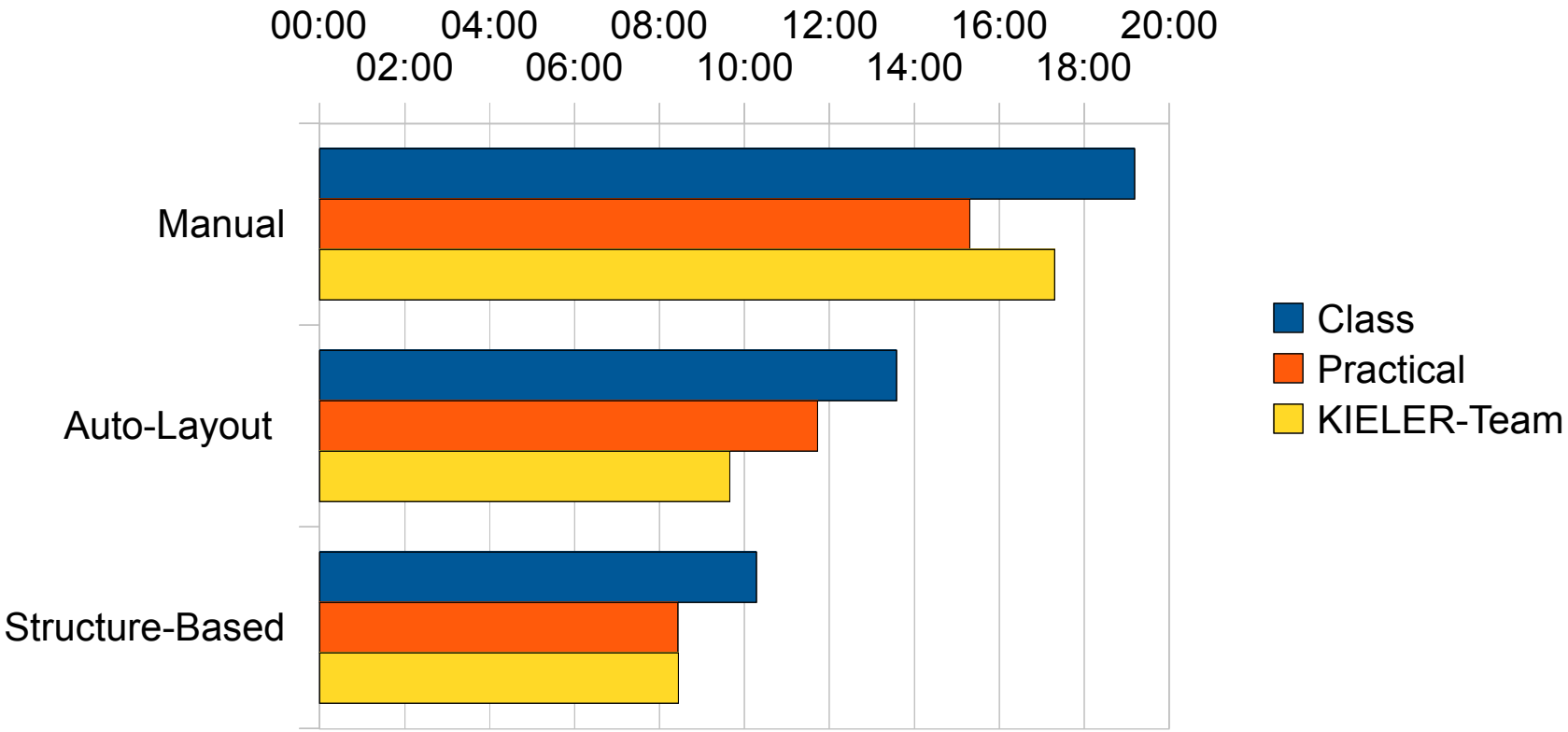
Pragmatics usually concentrates on practical aspects of how constructs and features of a language may be used to achieve various objectives (*e. g.*, when to use an assignment).

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

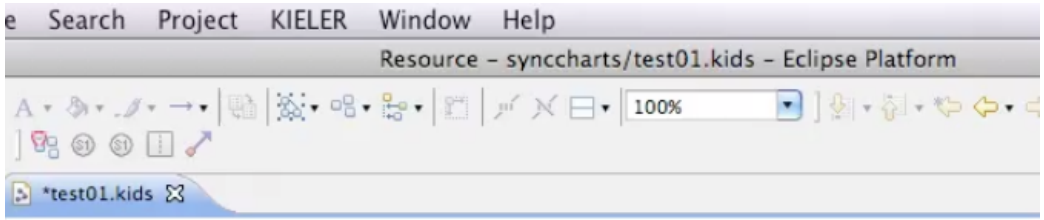
Editing Efficiency: An Experiment

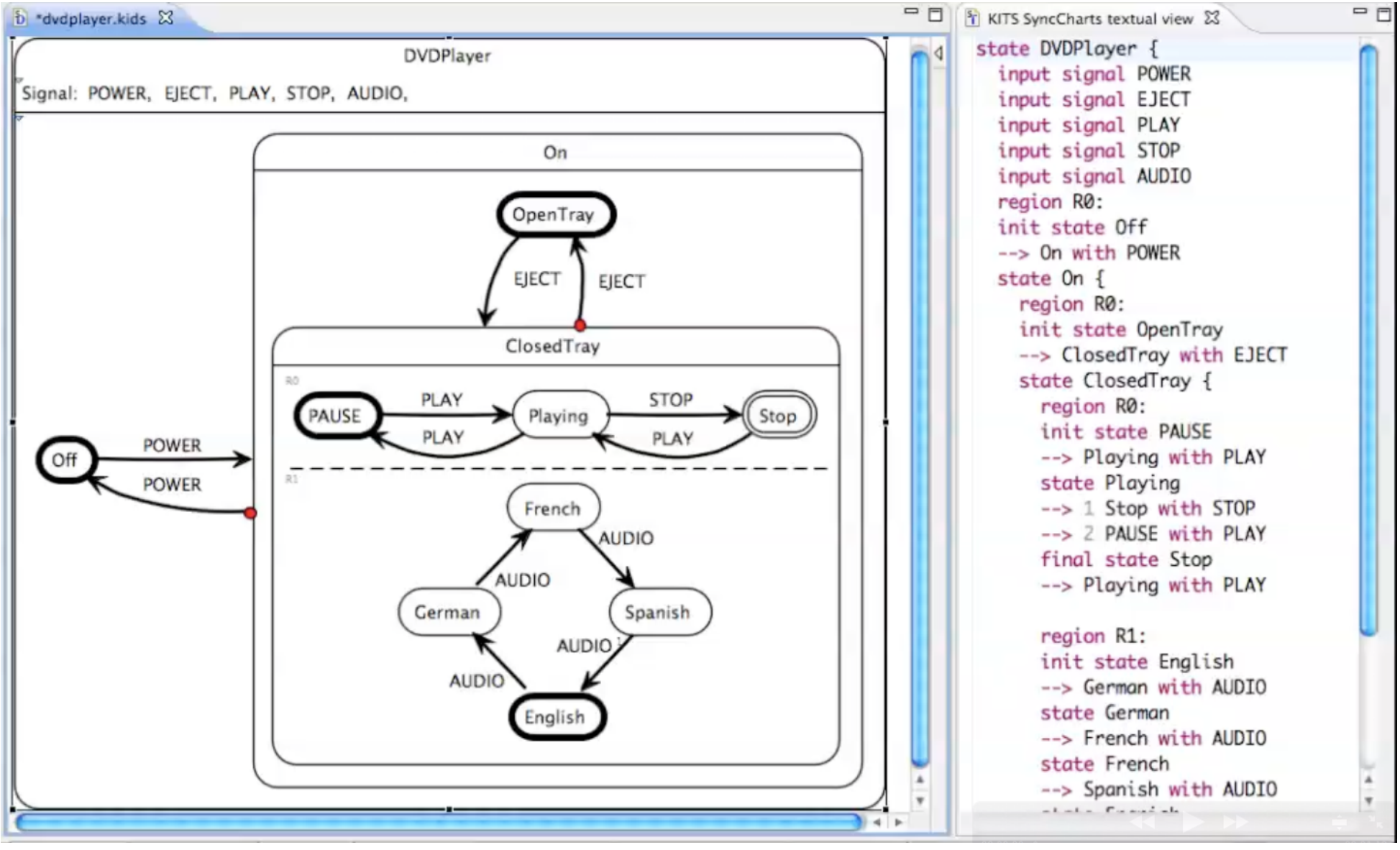
Task: create diagram from textual specification



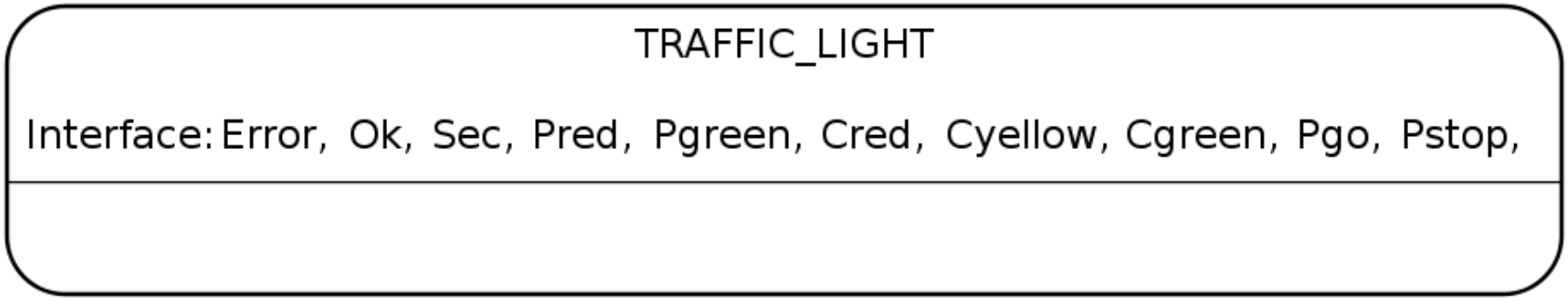
Build upon Layout: View Management

- Structure-Based Editing
- Textual Editing
- Simulation

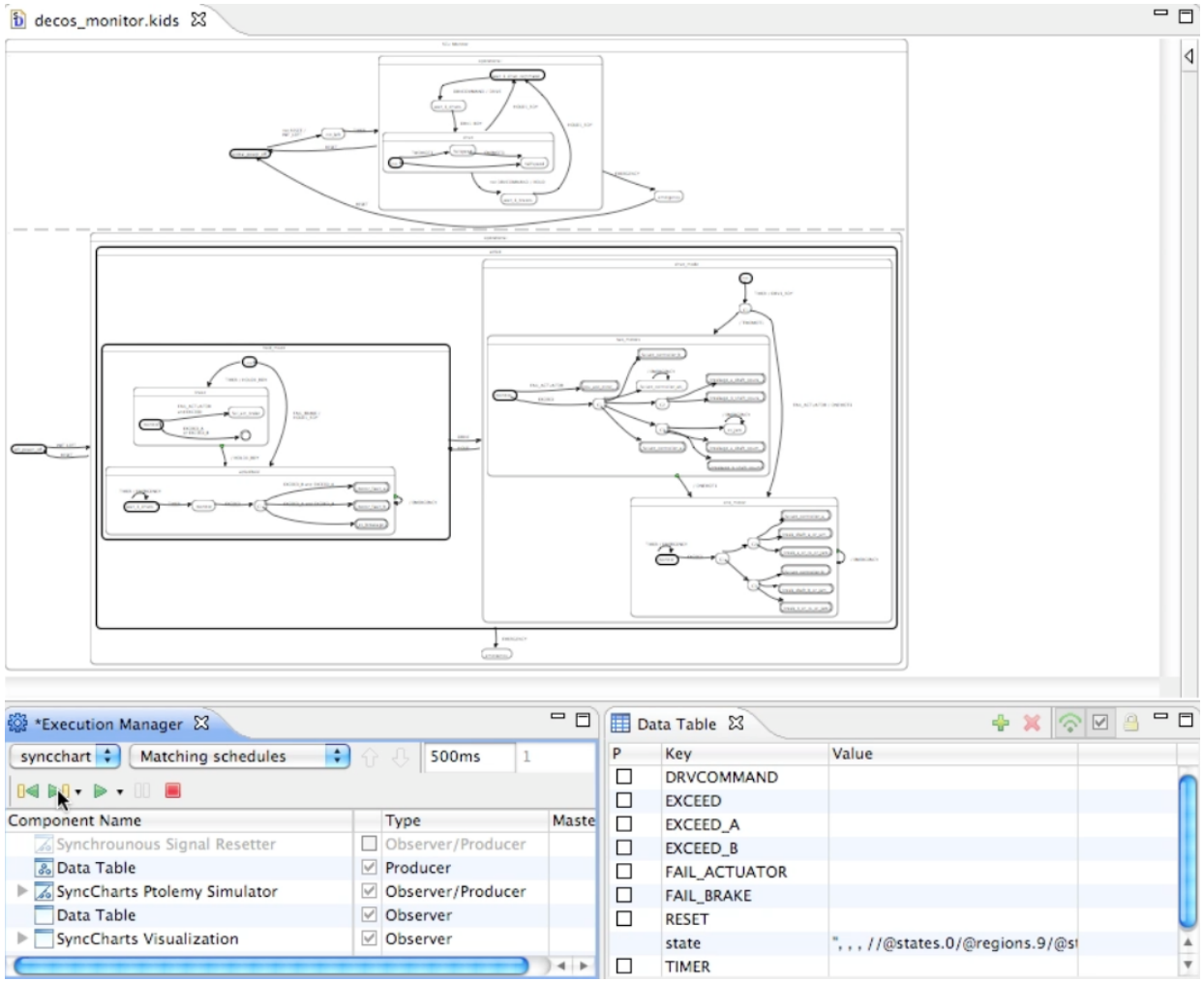

e Search Project KIELER Window Help
Resource - syncharts/test01.kids - Eclipse Platform
100%
*test01.kids

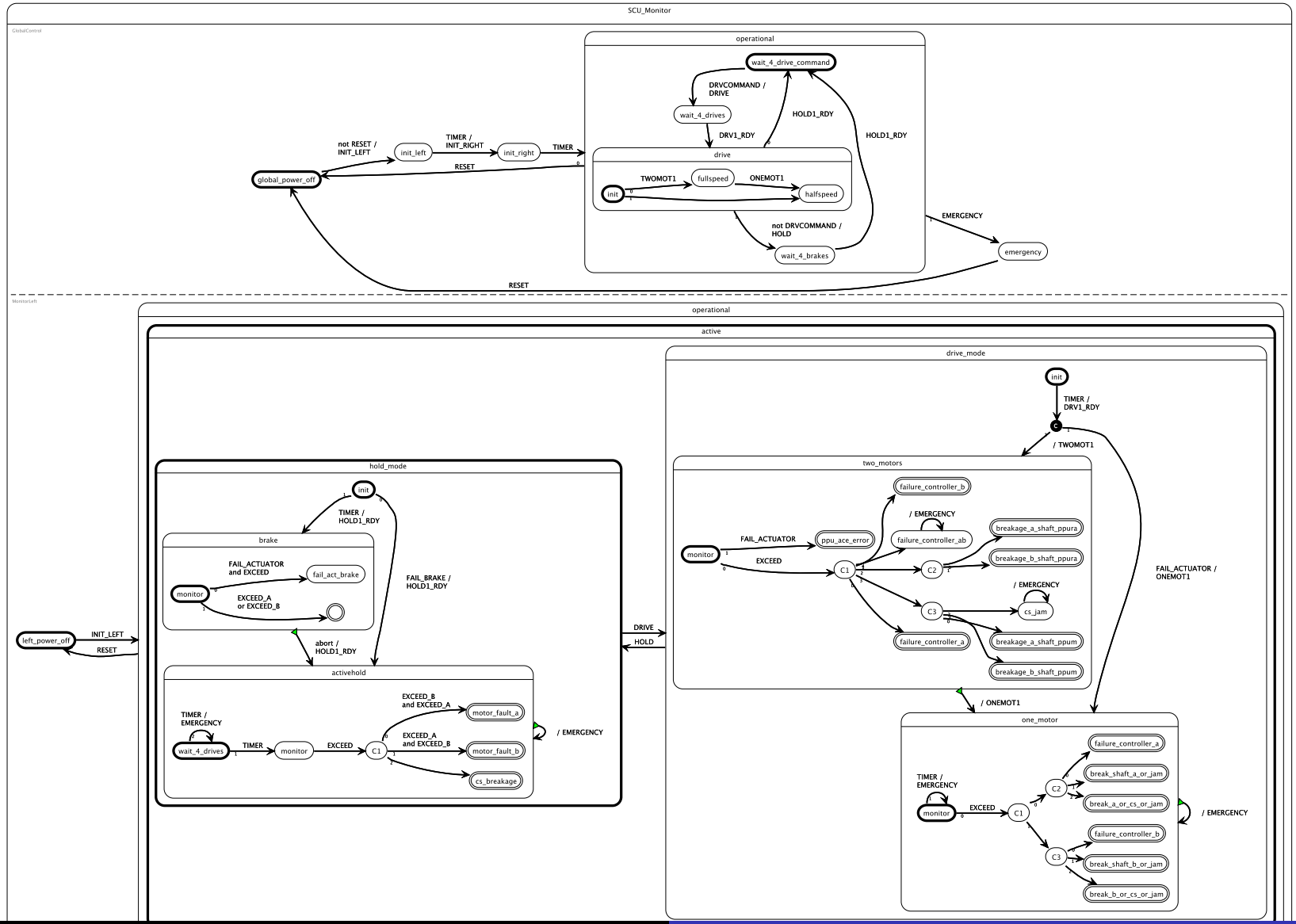

*dvdplayer.kids
DVDPlayer
Signal: POWER, EJECT, PLAY, STOP, AUDIO,
On
OpenTray
EJECT
EJECT
ClosedTray
R0
PAUSE
PLAY
PLAY
Playing
STOP
PLAY
Stop
Off
POWER
POWER
R1
French
AUDIO
AUDIO
German
Spanish
AUDIO
AUDIO
English
KITS SyncCharts textual view
state DVDPlayer {
input signal POWER
input signal EJECT
input signal PLAY
input signal STOP
input signal AUDIO
region R0:
init state Off
--> On with POWER
state On {
region R0:
init state OpenTray
--> ClosedTray with EJECT
state ClosedTray {
region R0:
init state PAUSE
--> Playing with PLAY
state Playing
--> 1 Stop with STOP
--> 2 PAUSE with PLAY
final state Stop
--> Playing with PLAY
region R1:
init state English
--> German with AUDIO
state German
--> French with AUDIO
state French
--> Spanish with AUDIO

Focus & Context

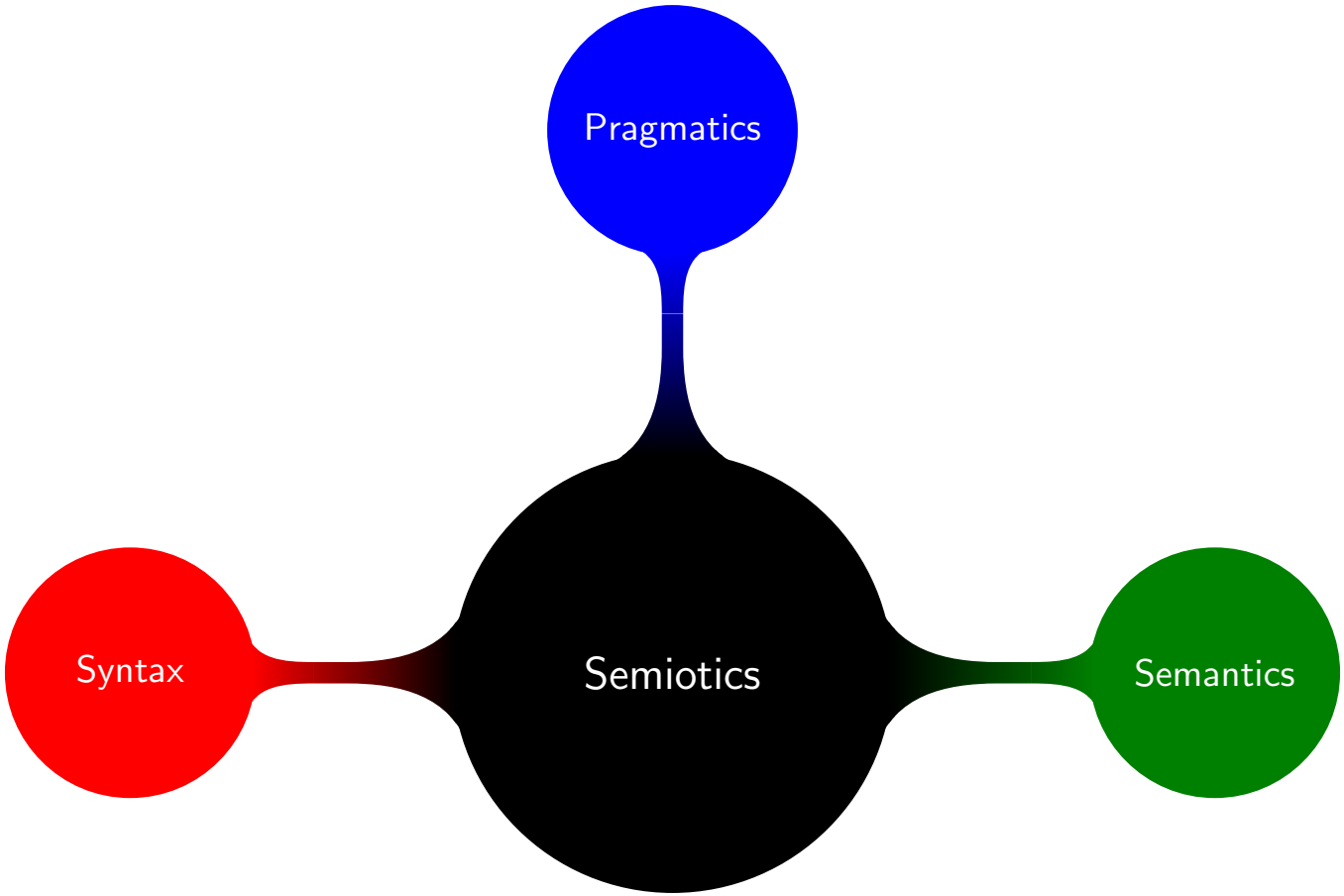


Simulation


decos_monitor.kids
*Execution Manager
syncchart
Matching schedules
500ms
Component Name
Type
Synchrounous Signal Resetter
Observer/Producer
Data Table
Producer
SyncCharts Ptolemy Simulator
Observer/Producer
Data Table
Observer
SyncCharts Visualization
Observer
Data Table
P
Key
Value
DRVCOMMAND
EXCEED
EXCEED_A
EXCEED_B
FAIL_ACTUATOR
FAIL_BRAKE
RESET
state
TIMER


SCU_Monitor
operational
wait_4_drive_command
DRVCOMMAND / DRIVE
wait_4_drives
DRV1_RDY
HOLD1_RDY
HOLD1_RDY
not RESET / INIT_LEFT
init_left
TIMER / INIT_RIGHT
init_right
TIMER
RESET
global_power_off
drive
TWOMOT1
fullspeed
ONEMOT1
halfspeed
init
not DRVCOMMAND / HOLD
wait_4_brakes
EMERGENCY
emergency
RESET
operational
active
drive_mode
init
TIMER / DRV1_RDY
/ TWOMOT1
two_motors
failure_controller_b
/ EMERGENCY
FAIL_ACTUATOR
ppu_ace_error
failure_controller_ab
breakage_a_shaft_ppura
monitor
EXCEED
C1
C2
breakage_b_shaft_ppura
FAIL_ACTUATOR / ONEMOT1
/ EMERGENCY
C3
cs_jam
failure_controller_a
breakage_a_shaft_ppum
breakage_b_shaft_ppum
/ ONEMOT1
one_motor
failure_controller_a
TIMER / EMERGENCY
break_shaft_a_or_jam
C2
break_a_or_cs_or_jam
monitor
EXCEED
C1
/ EMERGENCY
failure_controller_b
C3
break_shaft_b_or_jam
break_b_or_cs_or_jam
hold_mode
init
TIMER / HOLD1_RDY
brake
FAIL_ACTUATOR and EXCEED
fail_act_brake
monitor
EXCEED_A or EXCEED_B
FAIL_BRAKE / HOLD1_RDY
abort / HOLD1_RDY
left_power_off
INIT_LEFT
RESET
DRIVE
HOLD
activehold
EXCEED_B and EXCEED_A
motor_fault_a
TIMER / EMERGENCY
EXCEED_A and EXCEED_B
/ EMERGENCY
wait_4_drives
TIMER
monitor
EXCEED
C1
motor_fault_b
cs_breakage

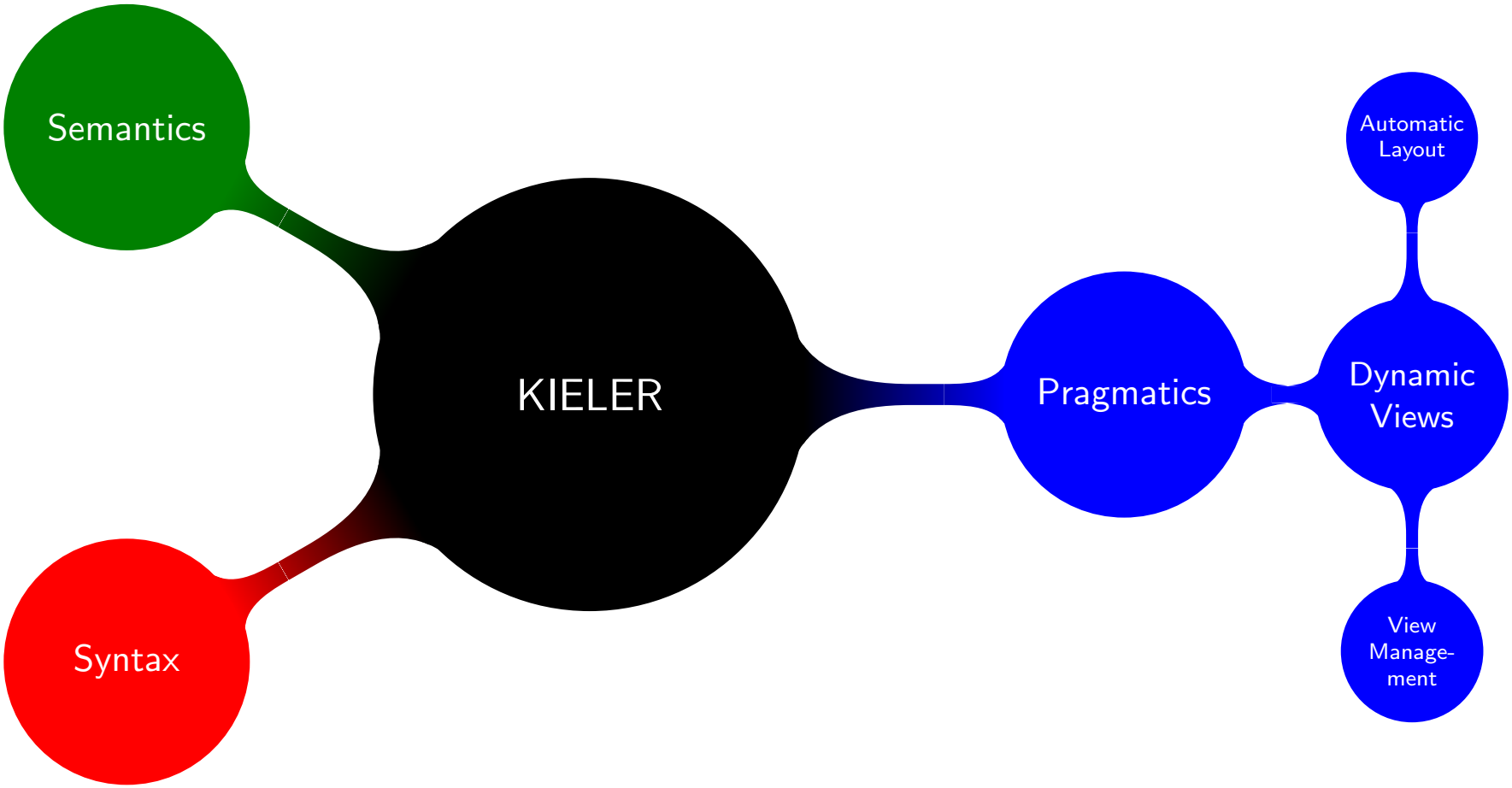
Semiotics



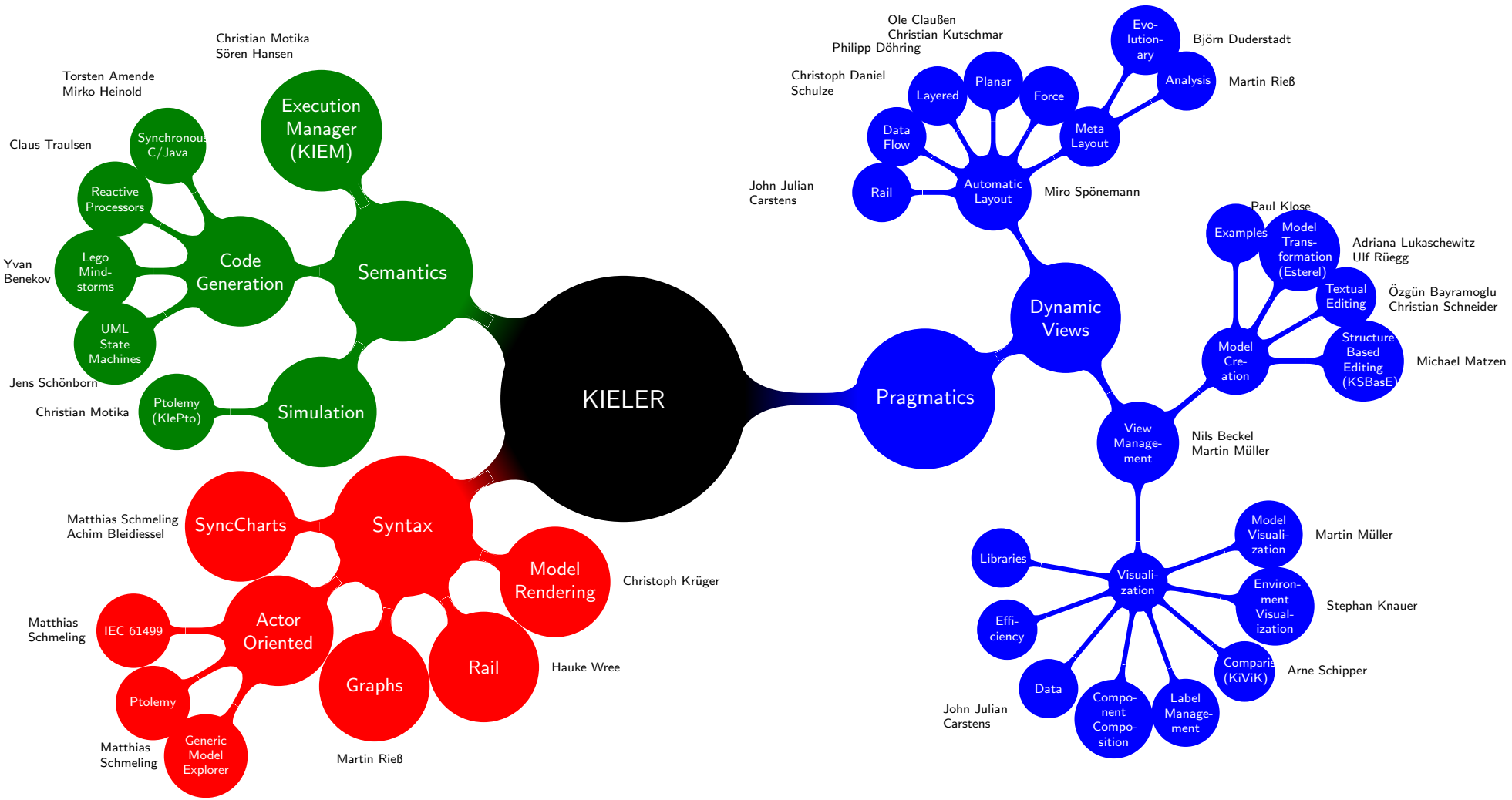
KIELER Objectives

- Free user of manual mechanical work.
  - Manual placing of graphical objects.
  - Manual navigation in complex models.
- Focus on pragmatics.
  - New interaction methodologies.
  - New analysis methodologies.
  - New ways to synthesize models.

KIELER Semiotics



KIELER Semiotics



Putting KIELER to Work