Light-Weight Synthesis of Ptolemy Diagrams
with KIELER

Ulf Rüegg, Christian Schneider, Christoph Daniel Schulze,
Miro Spönnemann, Christian Motika, and Reinhard von Hanxleden

Real-Time Systems and Embedded Systems Group
Department of Computer Science
Christian-Albrechts-Universität zu Kiel, Germany

10th Ptolemy Miniconference
Berkeley, 7 Nov. 2013
This model illustrates the well-known Brock-Ackerman anomaly. The two composite actors ActorA and ActorB implement exactly the same (non-deterministic) input/output relation. That is, given any two input sequences at the two input ports, the possible output sequences from each actor are the same. However, when wired as shown into two feedback loops, the two actors do not behave the same way. In particular, the upper feedback loop has more possible outputs than the bottom one.
Graphical Modeling

- 😊 Short learning curve (palette, Drag&Drop)
Graphical Modeling

- ☺ Short learning curve (palette, Drag&Drop)
- ☺ Readability (inspecting/understanding, mental map)
Graphical Modeling

- Short learning curve (palette, Drag&Drop)
- Readability (inspecting/understanding, mental map)
- Visualization of dynamics: Simulation
Graphical Modeling

- Short learning curve (palette, Drag&Drop)
- Readability (inspecting/understanding, mental map)
- Visualization of dynamics: Simulation
- Validation (detect obvious model errors)
Graphical Modeling (cont’d)

- Widely used in today’s industrial tool chains (e.g., SCADE) and academia (e.g., Ptolemy)
Graphical Modeling (cont’d)

- Widely used in today’s industrial tool chains (e.g., SCADE) and academia (e.g., Ptolemy)

- 😞 Readability (overview gets lost quickly)
Graphical Modeling (cont’d)

- Widely used in today’s industrial tool chains (e.g., SCADE) and academia (e.g., Ptolemy)
- 😞 Readability (overview gets lost quickly)
- 😞 Maintenance (requires lots of manual effort)
Graphical Modeling (cont’d)

▶️ Readability (layout critical for understanding semantics)
Graphical Modeling (cont’d) - Model Browsing

There is a corresponding deployment model that models the same traffic light control algorithm with wireless communication between the car light and the pedestrian light. Changes in the normal operation logs of either light here will be reflected automatically in the deployment model.

This model illustrates a typical design pattern where the top level is a SR model of the physical environment for a system under design. The read-down, down is a model model fashioned after the statechart model at the right. Open the TrafficLight editor to see how this is implemented.

In order to simulate correct and erroneous behavior, we use a simulator FMU to test between normal and abnormal cases.
Graphical Modeling (cont’d) - Model Browsing

Top-level model of the traffic light controller where there are two states, an error state and a normal state. Look inside the states to see the implementations.

Note that we are following the design of the Statecharts model shown on the top level, but there is a flaw in that design that shows up when constructing a deployment model. The flaw is that the Error and Ok states are at the top level, and internally contain concurrent operations of the car light and the pedestrian light. It should be the other way around. The car light and pedestrian light should be concurrent, and should internally each have Error and Ok states. This way, the car light and pedestrian light can be deployed in separate hardware.
Graphical Modeling (cont’d) - Model Browsing

The NormaC actor generates the control signals for the car stoplights under normal operating conditions. The NormaP actor reacts to these controls to generate the control signals for the pedestrian lights. Look inside each actor to see its implementation.

The CarLightNormal and PedestrianLightNormal actors here are instances of actor-oriented classes defined in other files. If you open the actors, you will open the other files. If you change the design, then all other instances of this class will see the change. In particular, the WirelessDeployment example uses the same instances.

Top-level model where there are normal states. No implementations.

Note that we are following top level, but there is a deployment model level, and internally optional concurrent operations on the car light and the pedestrian light. It should be otherwise around. The car light and pedestrian light should be concurrent, and should internally each have Error and Ok states. This way, the car light and pedestrian light can be deployed in separate hardware.
Graphical Modeling (cont’d) - Model Browsing

The NormalC actor generates the control signals for the car stoplights under normal operating conditions. The NormaP actor reacts to these controls to generate the control signals for the pedestrian lights. Look inside each actor to see its implementation.

Note that we are following a top-level model, which is a deployment model, and internally contain concurrent operation. The pedestrian light should be concurrent, and should contain internal states. This way, the car light and pedestrian light are separate hardware.
Graphical Modeling (cont’d) - Model Browsing

The NormaC actor generates the control signals for the car stoplights under normal operating conditions. The NormaP actor reacts to these controls to generate the control signals for the pedestrian lights. Look inside each actor to see its implementation.

Note that we have top-level, deployment, and internal levels. The pedestrian light should have separate hardware.
Graphical Modeling (cont’d) - Model Browsing

▶ Fokus&Context, inner and outer ports vs. performance
Graphical Modeling vs. Textual Modeling

- Emerging trend: *Textual Modeling*
  - Concrete syntax is text
Graphical Modeling vs. Textual Modeling

- Emerging trend: *Textual Modeling*
  - Concrete syntax is text
  - 😊 Rapid Development (good tooling support, e.g., Xtext)
Graphical Modeling vs. Textual Modeling

- **Emerging trend:** *Textual Modeling*
  - Concrete syntax is text
  - **😊** Rapid Development (good tooling support, e.g., Xtext)
  - **😊** Handling (model diffs, version control, tool interchange)
Graphical Modeling vs. Textual Modeling

- Emerging trend: *Textual Modeling*
  - Concrete syntax is text
  - 😊 Rapid Development (good tooling support, e.g., Xtext)
  - 😊 Handling (model diffs, version control, tool interchange)
  - 😊 Readability, Maintenance (formatter)
Graphical Modeling vs. Textual Modeling

- Emerging trend: *Textual Modeling*
  - Concrete syntax is text
  - 😊 Rapid Development (good tooling support, e.g., Xtext)
  - 😊 Handling (model diffs, version control, tool interchange)
  - 😊 Readability, Maintenance (formatter)
  - 😞 Readability (harder to inspect a larger model, mental map)
**Graphical Modeling vs. Textual Modeling**

- **Emerging trend:** *Textual Modeling*
  - Concrete syntax is text
  - 😊 Rapid Development (good tooling support, e.g., Xtext)
  - 😊 Handling (model diffs, version control, tool interchange)
  - 😊 Readability, Maintenance (formatter)
  - 😞 Readability (harder to inspect a larger model, mental map)
  - 😞 Validation (harder to see model errors)
Graphical Modeling vs. Textual Modeling

- Emerging trend: *Textual Modeling*
  - Concrete syntax is text
  - 😊 Rapid Development (good tooling support, e.g., Xtext)
  - 😊 Handling (model diffs, version control, tool interchange)
  - 😊 Readability, Maintenance (formatter)
  - 😞 Readability (harder to inspect a larger model, mental map)
  - 😞 Validation (harder to see model errors)
  - 😊 Visualization of dynamics: Simulation
Contribution

- Get all benefits from graphical modeling
Contribution

- Get all benefits from graphical modeling
- Preserve all the benefits from textual modeling
Contribution

- Get all benefits from graphical modeling
- Preserve all the benefits from textual modeling
- \(\Rightarrow\) Automatic synthesis of diagrams: KIELER Light-Weight Diagram (KLighD)
Overview

- Foundations & Concept
- Demo
- Case Study Results
Synthesis of Diagrams

- Input: Model (dsl, xml, moml)

[C. Schneider et al., VL/HCC’13]
Synthesis of Diagrams

- Input: Model (dsl, xml, moml)
- Output: Configurable diagram

[C. Schneider et al., VL/HCC’13]
Synthesis of Diagrams

- **Input:** Model (dsl, xml, moml)
- **Output:** Configurable diagram
  - Diagram options: E.g., show transition labels

[C. Schneider et al., VL/HCC’13]
Synthesis of Diagrams

- **Input:** Model (dsl, xml, moml)
- **Output:** Configurable diagram
  - Diagram options: E.g., show transition labels
  - Layout options: E.g., direction or spacing

[C. Schneider et al., VL/HCC’13]
Synthesis of Diagrams

- Input: Model (dsl, xml, moml)
- Output: Configurable diagram
  - Diagram options: E.g., show transition labels
  - Layout options: E.g., direction or spacing
- Requirement: Automatic Layout (→ KIML)

[C. Schneider et al., VL/HCC’13]
KLighD Architecture

[C. Schneider et al., VL/HCC’13]
KLighD Demo
Ptolemy Case Study: Model Browsing

- ![Diagram](image.png)

- **Readability**: (normalized diagrams with fixed layout settings, configurable settings)
Ptolemy Case Study: Model Browsing

- 😊 Readability (normalized diagrams with fixed layout settings, configurable settings)
- 😊 Hierarchy (no new windows, inner and outer ports)
Readability (normalized diagrams with fixed layout settings, configurable settings)
- Hierarchy (no new windows, inner and outer ports)
- Large models (Focus&Context, collapse & expand)
Ptolemy Case Study: Model Browsing

- ☺ Readability  (normalized diagrams with fixed layout settings, configurable settings)
  - ☻ Hierarchy  (no new windows, inner and outer ports)
  - ☻ Large models  (Focus&Context, collapse & expand)
  - ☻ Complex models  (filter details, e.g., transition labels)
Ptolemy Case Study: Model Browsing

- Readability (normalized diagrams with fixed layout settings, configurable settings)
- Hierarchy (no new windows, inner and outer ports)
- Large models (Focus&Context, collapse & expand)
- Complex models (filter details, e.g., transition labels)
- Maintenance / Handling (create/edit the model in Vergil, generate it, use a textual DSL, ...)
Ptolemy Case Study: Model Browsing

- ☑️ Readability  (normalized diagrams with fixed layout settings, configurable settings)
  - ☑️ Hierarchy  (no new windows, inner and outer ports)
  - ☑️ Large models  (Focus&Context, collapse & expand)
  - ☑️ Complex models  (filter details, e.g., transition labels)

- ☑️ Maintenance / Handling  (create/edit the model in Vergil, generate it, use a textual DSL, ...)
- ☑️ Light-Weight  (no editing, no transactions → just transient views)
Model Editing & Simulation
Model Editing & Simulation

Light-Weight Synthesis of Ptolemy Diagrams with KIELER
Summary and Outlook

- Models are created once but read many times
- Large and complex, hierarchical models are hard to read and maintain
Summary and Outlook

- Models are created once but read many times
- Large and complex, hierarchical models are hard to read and maintain
- Automatic light-weight diagrams
  - help browsing/reading
  - and maintaining models
Summary and Outlook

- Models are created once but read many times
- Large and complex, hierarchical models are hard to read and maintain
- Automatic light-weight diagrams
  - help browsing/reading
  - and maintaining models
- Models can be textual or graphical
To Go Further

**Klauske, L. K., Schulze, C. D., Spönenmann, M., and von Hanxleden, R.**

Improved layout for data flow diagrams with port constraints.

**Schneider, C., Spönenmann, M., and von Hanxleden, R.**

Just model! – Putting automatic synthesis of node-link-diagrams into practice.
In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC’13)* (San Jose, CA, USA, 15–19 Sept. 2013).
With accompanying poster.

**Sugiymama, K., Tagawa, S., and Toda, M.**

Methods for visual understanding of hierarchical system structures.

**UNI Kiel, Real-Time and Embedded Systems Group.**

KIELER webpage.
http://www.informatik.uni-kiel.de/en/rtsys/kieler/.

**Von Hanxleden, R., Lee, E. A., Motika, C., and Fuhrmann, H.**

Multi-view modeling and pragmatics in 2020 — position paper on designing complex cyber-physical systems.

Thank you for your attention and participation!

Any questions or suggestions?
Ptolemy Case Study: Model Browsing

[Image of a Ptolemy model browsing interface]
Ptolemy Case Study: Model Browsing (cont’d)
General Data Structure Visualization
KLighDning - Collaborative Browser-Based Viewer