Taming Graphical Modeling

On Pragmatics-Aware Model-Driven Engineering

Reinhard von Hanxleden

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

MDE Workshop
HPI, Potsdam, May 30, 2014
Pragmatics of Model-Based Design

Pragmatics: relation of signs to their users
Pragmatics of Model-Based Design

Pragmatics: relation of signs to their users

Syntax: relations between signs
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
**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)
Modeling pragmatics =
def practical issues of model creation, editing, browsing, versioning, . . .
Pragmatics-aware modeling =
def model-driven engineering with specific support for designer productivity
Modeling pragmatics $=_{\text{def}}$ practical issues of model creation, editing, browsing, versioning, ...
Modeling pragmatics $=_{\text{def}}$ practical issues of model creation, editing, browsing, versioning, \ldots
Pragmatics-aware modeling $=_{\text{def}}$ model-driven engineering with specific support for designer productivity
Pragmatics-Unaware Modeling

Context missing
Pragmatics-Unaware Modeling
Quickly loose details
Pragmatics-Unaware Modeling
Lack of overview, tedious window management

This model illustrates a typical design pattern where the top level is a DE model of the physical environment for a system under design. The next level down is a modal model fashioned after the statecharts model on the right. Open the TrafficLight actor to see how it is implemented.

The PoisonClock actor occasionally injects an Error signal. The Error condition then lasts 5 seconds, as determined by the TimedDelay actor.

Authors: Reinhard von Hanxleden, Huining Feng, and Edward A. Lee
Pragmatics-Unaware Modeling
Lack of overview, tedious window management

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.
Pragmatics-Unaware Modeling

Lack of overview, tedious window management
Pragmatics-Unaware Modeling
Lack of overview, tedious window management

This state machine controls the car lights. It uses the count variable to stay red for three seconds and to stay green for two.
Pragmatics-Unaware Modeling

Lack of overview, tedious window management

This model turns the pedestrian lights green when the car control lights go red.
Pragmatics-Unaware Modeling
Lack of overview, tedious window management

This model just blinks the yellow lights on the car control and turns off the pedestrian lights.
Pragmatics-Unaware Modeling

Better overview, but high manual effort
Pragmatics-Unaware Modeling

Diagrams difficult to read
Pragmatics-Unaware Modeling

Diagrams difficult to read
Pragmatics-Unaware Modeling

Diagrams difficult to read
Pragmatics-Unaware Modeling
Overloaded screen real-estate
Pragmatics-Unaware Modeling

Painful editing technology
Pragmatics-Unaware Modeling

Manual effort increases significantly with model complexity
Pragmatics-Unaware Modeling

Manual effort increases significantly with model complexity
Are Textual DSLs the Answer?

- Editing text much less tedious than editing graphics
- Revision control simpler
- Portable, tool independent
Still Want Graphical Views!

- Text requires string matching to uncover structure
- Diagram makes structure obvious
- Unreachable state in model emerges
Still Want Graphical Views!

- Diagram uncovers specification flaw (upper right corner)
Pragmatics-Aware Modeling

Free user of tedious mechanical work, such as . . .

- manual placing of graphical objects
- manual navigation in complex models
Pragmatics-Aware Modeling

Free user of tedious mechanical work, such as . . .

- manual placing of graphical objects
- manual navigation in complex models

Focus on **pragmatics**:

- New interaction methodologies.
- New analysis methodologies.
- New ways to synthesize models.
Pragmatics-Aware Modeling

Free user of tedious mechanical work, such as . . .
- manual placing of graphical objects
- manual navigation in complex models

Focus on pragmatics:
- New interaction methodologies.
- New analysis methodologies.
- New ways to synthesize models.

Our experimental platform: KIELER
Kiel Integrated Environment
for Layout Eclipse RichClient
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.
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*.
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.
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.

A Key to Pragmatics: MVC

The Model
Synthesis & Editing
- Structure-based editing
- Synthesis
  - Dual-modeling / Multi-modeling
  - Textual modeling
- Scaling
- Patterns
- Product lines

The View
Representing the model
- Automatic layout
- Filtering (incl. label management)
- Focus & context, zooming, panning
- Morphing
- Data visualization

The Controller
Interpreting the model
- Interfacing to other modeling tools
- Correctness check, static analysis
- Visual differencing
- Simulation engine

Fuhrmann, von Hanxleden
On the Pragmatics of Model-Based Design
Overview

Pragmatics-Aware Modeling
Definition and Motivation
MVC a Key to Pragmatics!

Pragmatics/KIELER Spotlights
KIELER Infrastructure for Meta Layout (KIML)
An Experiment
KIELER Lightweight Diagrams (KLighD)

Wrap-Up
Diagram Editor View

KIELER Infrastructure for Meta Layout

KIELER Infrastructure for Meta Layout (KIML)
An Experiment
KIELER Lightweight Diagrams (KLighD)

Diagram Editor View

KIELER Infrastructure for Meta Layout

Layout Algorithm

Schulze, Spöne, von Hanxleden

Drawing Layered Graphs with Port Constraints
JVLC'14
Diagram Editor View

KIELER Infrastructure for Meta Layout

KGraph

Layout Algorithm

- GMF
- Graphiti
- Papyrus
- Yakindu

- Graphviz (Dot, Neato, FDP, Twopi, Circo)
- Open Graph Drawing Framework (OGDF)
  (Layer-based, Planarization, Force-directed)
- Own Implementations (Data flow diagrams)
GMF
Graphiti
Papyrus
Yakindu

Graphviz (Dot, Neato, FDP, Twopi, Circo)
Open Graph Drawing Framework (OGDF)
(Layer-based, Planarization, Force-directed)
Own Implementations (Data flow diagrams)

Schulze, Spönnemann, v. Hanxleden
Drawing Layered Graphs with Port Constraints
JVLC’14
Editing Efficiency: An Experiment

**Task:** create diagram from textual specification
Editing Efficiency: An Experiment

Task: create diagram from textual specification

Fuhrmann, von Hanxleden
Taming Graphical Modeling
MODELS’10
KIELER Lightweight Diagrams (KLighD)
KIELER Lightweight Diagrams (KLighD)

```plaintext
state machine {
    input signal POWER
    region:
        init state Off
        --> On with POWER;
        state On;
}
```
KIELER Lightweight Diagrams (KLighD)

state machine {
    input signal POWER
    region:
        init state Off
        --> On with POWER;
    state On;
}

KLighD

diagram
This model shows a simple adaptive cruise control system, illustrating model-integrated control strategies. A leading car model produces information that is observed with possible flaws by a following car. If the following car detects flaws, it uses a conservative strategy. Otherwise, it tracks the leading car closely.

Simulate a wireless network that corrupts the data when the fault input is true. Simulate a car that attempts to detect faults in communication and adapt its behavior.

Author: Xiaojun Liu and Edward A. Lee
Ptolemy Browser (Based on KLighD)
Lightweight Diagrams: Building blocks

- Automatic arrangement of depicted elements – macro layout
- Local arrangement of the figures’ primitives – micro layout
- A description language to formulate diagrams smartly
Lightweight Diagrams: Building blocks

- Automatic arrangement of depicted elements – **macro layout**
- Local arrangement of the figures’ primitives – **micro layout**
- A description language to formulate diagrams smartly
Lightweight Diagrams: Building blocks

- Automatic arrangement of depicted elements – macro layout
- Local arrangement of the figures’ primitives – micro layout
- A description language to formulate diagrams smartly
Lightweight Diagrams: Building blocks

- Automatic arrangement of depicted elements – **macro layout**
- Local arrangement of the figures’ primitives – **micro layout**
- A description language to formulate diagrams smartly

![Diagram showing the transition from OpenTray to ClosedTray with PAUSE and PLAY actions.](image-url)
Lightweight Diagrams: Building blocks

- Automatic arrangement of depicted elements – macro layout
- Local arrangement of the figures’ primitives – micro layout
- A description language to formulate diagrams smartly

```kliml
KNode {
  width=200 height=200
KRoundedRectangle cornerWidth=10, cornerHeight=10 {
  KText "On" {
    KAreaPlacementData
topLeft TOP abs=0px rel=0, LEFT abs=10px rel=0
    bottomRight TOP abs=20px rel=0, RIGHT abs=10px rel=0
  }
}
KPolyline {
  points =
    (TOP abs=20px rel=0, LEFT abs=0px rel=0),
    (TOP abs=20px rel=0, RIGHT abs=0px rel=0)
}
...
Architecture overview
Architecture overview
Architecture overview

transformation A  
transformation B  
transformation ...

model  

KLighD  

KIML  

diagram
Architecture overview
Architecture overview

transformation A

transformation B

transformation ...

1) model

2) dispatch

KLighD

KIML

model

diagram
Architecture overview

transformation A
transformation B
transformation ...

1) model
2) dispatch
3) model

transformations A, B, and ... connected to KLighD, which is connected to KIML and a diagram.
Architecture overview

1) model
2) dispatch
3) model
4) KGraph & KRendering
transformation A
transformation B
transformation ...

KIML
KLighD

diagram
Architecture overview

1) model

2) dispatch

3) model

4) KGraph & KRendering

5) KGraph & KRendering data

transformation A

transformation B

transformation ...

model

KLighD

KIML

diagram
Architecture overview
Architecture overview

1) model
2) dispatch
3) model
4) KGraph & KRendering
5) KGraph & KRendering data
6) KGraph, KRendering & layout data
7) KGraph, KRendering & layout data

KIML
KLighD
transformation A
transformation B
transformation ...

KIML Infrastructure for Meta Layout (KIML)
An Experiment
KIELER Lightweight Diagrams (KLighD)
KGraph/KRendering Diagram Notation Format

- Designed for describing concrete diagrams
  - Formulated in EMF’s meta modeling language *Ecore*
  - Lifts diagram synthesis to modeling level of abstraction
KGraph/KRendering Diagram Notation Format

- Designed for describing concrete diagrams
  - Formulated in EMF’s meta modeling language *Ecore*
  - Lifts diagram synthesis to modeling level of abstraction
- Clear separation of diagram structure (KGraph, coordinates) and appearance (KRendering)
- Enables declarative micro layout statements
- Aims at reducing concrete-coordinate-related statements as much as possible

Schneider, Spönenmann, von Hanxleden
*Just Model! – Putting Automatic Synthesis of Node-Link-Diagrams into Practice*
VL/HCC’13
Outline

Pragmatics-Aware Modeling
Definition and Motivation
MVC a Key to Pragmatics!

Pragmatics/KIELER Spotlights
KIELER Infrastructure for Meta Layout (KIML)
An Experiment
KIELER Lightweight Diagrams (KLighD)

Wrap-Up
Wrap-Up

The Problem

- Lots of productivity wasted with drawing diagrams manually
- Sign for trouble: the palette
Wrap-Up

The Problem
- Lots of productivity wasted with drawing diagrams manually
- Sign for trouble: the palette

Our Approach
- Pragmatics-aware modeling
Wrap-Up

The Problem

▶ Lots of productivity wasted with drawing diagrams manually
▶ Sign for trouble: the palette

Our Approach

▶ Pragmatics-aware modeling
▶ Let designer concentrate on *model*, automatically synthesize *views*
Wrap-Up

The Problem
- Lots of productivity wasted with drawing diagrams manually
- Sign for trouble: the palette

Our Approach
- Pragmatics-aware modeling
- Let designer concentrate on model, automatically synthesize views
- Employ filtering to synthesize views customized to user stories
Wrap-Up

The Problem

- Lots of productivity wasted with drawing diagrams manually
- Sign for trouble: the palette

Our Approach

- Pragmatics-aware modeling
- Let designer concentrate on *model*, automatically synthesize *views*
- Employ *filtering* to synthesize views customized to user stories
- Key enabler: *automatic layout*
Outlook

- DSL for synthesis of transient views
- Smart layout configuration
- Collaborative web-based model browsing
- Efficient browsing of very complex models (think google maps)
Get In Touch

- Open-source license (EPL) encourages commercial use
- http://www.informatik.uni-kiel.de/rtsys/kieler/
Get In Touch

- Open-source license (EPL) encourages commercial use
- http://www.informatik.uni-kiel.de/rtsys/kieler/

thanks!

questions or comments?
Appendix
Further examples

Component clustering

Christoph Blees
Eine Methode zur Entwicklung modularer Produktfamilien
Further examples

Net lists
Further examples

Control flow graph
Further examples

Domain-specific components
Further examples

Domain-specific components
Further examples

Domain-specific components
Related Work

- Robert Ian Bull
  Towards A Model Driven Engineering Approach For Information Visualization

- Jan Koehnlein
  Discovery Diagrams for the Generic Graphical View
  http://koehnlein.blogspot.de/2012/01/discovery-diagrams-for-generic.html
Related Work

Robert Ian Bull
Towards A Model Driven Engineering Approach For Information Visualization

Jan Koehnlein
Discovery Diagrams for the Generic Graphical View
http://koehnlein.blogspot.de/2012/01/discovery-diagrams-for-generic.html

Martin Auer, T. Tschurtschenthaler, and Stefan Biffl,
A Flyweight UML Modelling Tool for Software Development in Heterogeneous Environments
http://www.umlet.com

OMG Diagram Definition (DD), 2012
http://www.omg.org/spec/DD/1.0/
More Related Work

Akos LedeÁEzi et al.
The Generic Modeling Environment
IEEE Workshop on Intelligent Signal Processing (WISP 2001)

Gergely Mezei et al.
Visual presentation solutions for domain specific languages
IASTED International Conference on Software Engineering (2006)

M. Minas
Generating meta-model-based freehand editors
3rd International Workshop on Graph Based Tools (GraBaTs’06)

Eclipse GMF (http://www.eclipse.org/modeling/gmp/)

Graphiti project (http://www.eclipse.org/graphiti/)

Benjamin B. Bederson et al.
Toolkit Design for Interactive Structured Graphics