Modelling Software!
Forget about Programming!

Ekkart Kindler
Department of Computer Science
University of Paderborn

Modelling Software!
Forget about Programming!

Ekkart Kindler
Technical University of Denmark
Informatics and Mathematical Modelling

Competence Centre for Model-based Software Engineering

Point of this talk

… a glimpse of how we could develop software in 10 years – without doing any programming at all.

Program vs. Software

is much more than

Software >> Program!

Software Engineering >>> Programming!

is much much more than

Programming … ering

Software
Software Engineer
Software Engineering

If somebody has built a garage, we would never let him built a sky scraper, would we?

But, if somebody knows to program, we let him do software engineering, don't we?

Program
Programmer
Programming

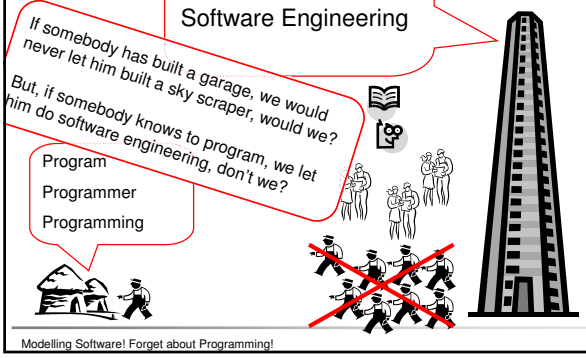
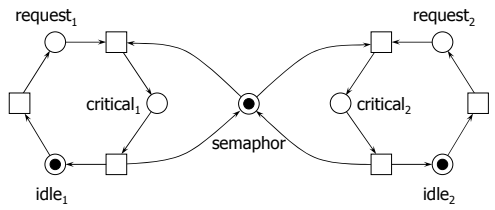


Table of Contents

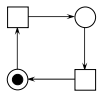
- Programming vs. Software Engineering
- Software Engineering Today
  - meta modelling & MOF
  - a Petri net editor in 15 minutes (EMF/GMF)
  - workflow management
- Status Quo
- Perspectives
  - modelling dynamic behaviour
  - modelling and integrating behaviour
  - defining transformations
- Theses

Example: A Petri Net (Model)

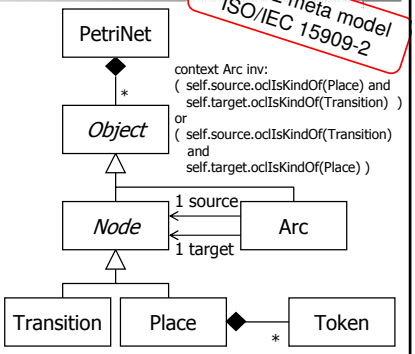


Models and Meta Models

→ PNML meta model ISO/IEC 15909-2

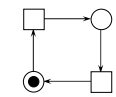


Petri net model



Meta model for Petri nets

Abstract and Concrete Syntax



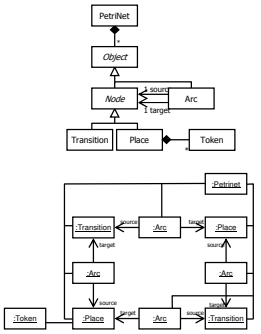
graphical / **concrete syntax**

Overview

meta model — build-time

is an instance of

model — runtime

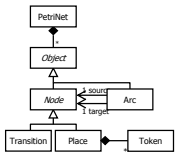


Benefits of Modelling

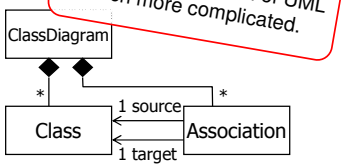
- Better understanding
- Mapping of instances to XML syntax (XMI)
- Automatic code generation
  - API for creating, deleting and modifying model
  - Methods for loading and saving models (in XMI)
  - Standard mechanisms for keeping track of changes (observers)

Class Diagrams are Models too

Note: The real model of UML is much more complicated.



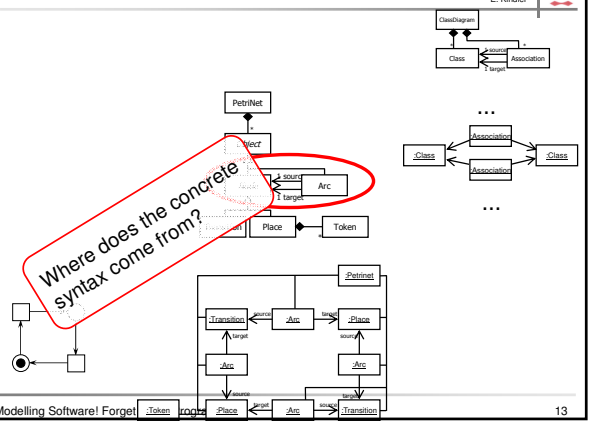
UML model



Meta model for UML (class diagrams)

The term meta model makes sense now!

Different Meta Levels: MOF

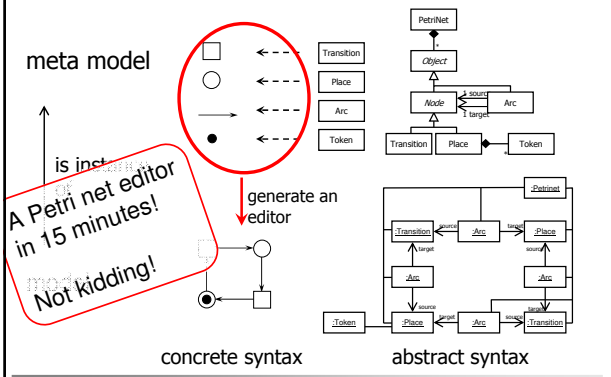


Answers:

- Prog...
- Standard technology for mapping abstract to concrete syntax: EMF / GMF / EMFT

Not a good answer in this talk!

EMF/GMF Technology (Eclipse)

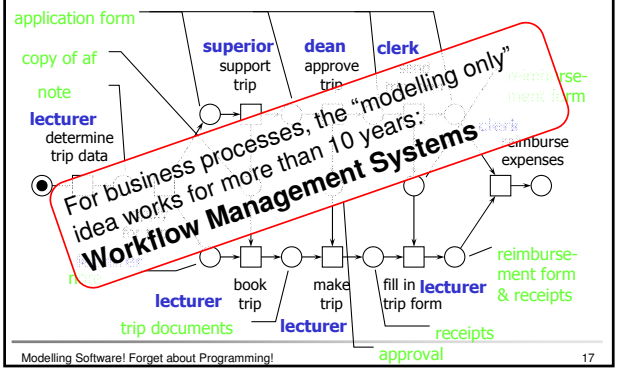


Benefits of Modelling (cntd.)

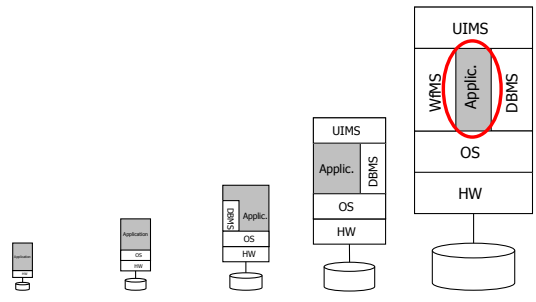
- Better Understanding
- Mapping of instanc... ...tax (XMI)
- A... ...de Generation
  - A... for creating, deleting and modifying model
  - Methods for loading and saving models (in XMI)
  - Standard mechanisms for keeping track of changes (observers)
  - Editors and GUIs

How about modelling business logic?

Example: Business Trip



A historical view (v.d.Aalst 96)



Status Quo

Where does the “modelling only” idea work?

Graphical editors

- graphics only
- standard functions only
- no business logic

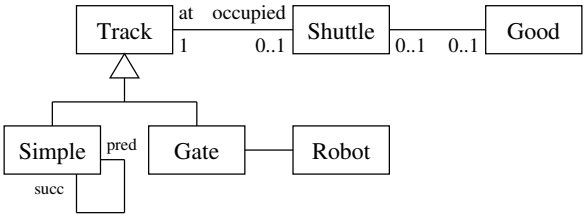
Business processes

- standard GUI only
- business logic explicitly modelled
- dedicated modelling notation

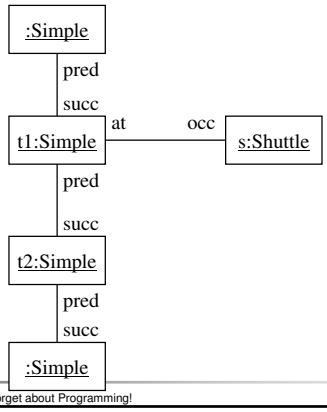
Table of Contents

- Programming vs. Software Engineering
- Software Engineering Today
  - meta modelling & MOF
  - a Petri net editor in 15 minutes (EMF/GMF)
  - workflow management
- Status Quo
- Perspectives
  - modelling dynamic behaviour
  - modelling and integrating behaviour
  - defining transformations
- Theses

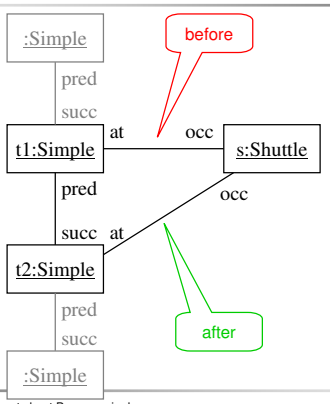
Example: Material Flow System



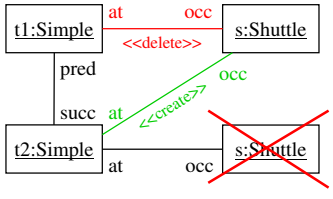
An Instance



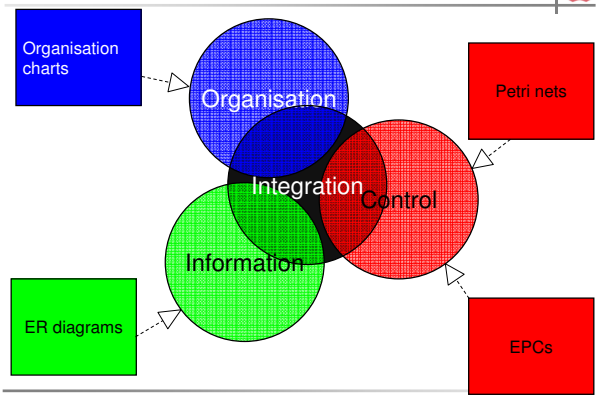
Behaviour



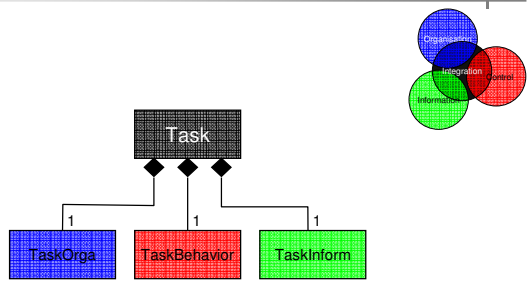
Story Pattern Defining Behaviour



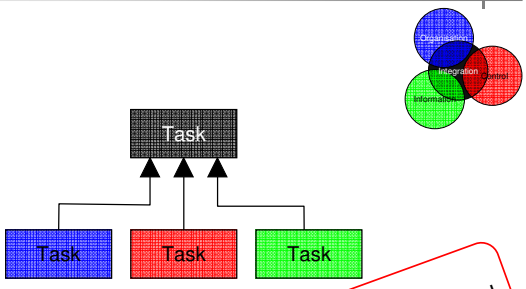
AMFIBIA



Aspect Oriented (Meta) Modelling

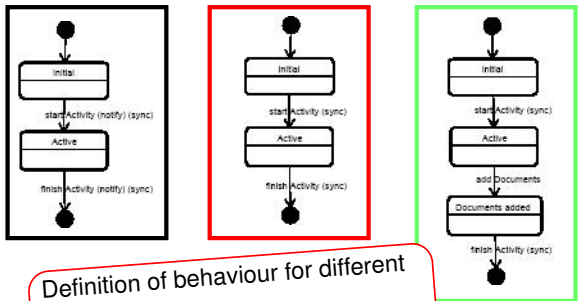


Aspect Orientation Modelling



AoM: Aspect Oriented Modelling

Modelling the Behaviour



Definition of behaviour for different aspects.
Coordination and integration of the behaviour of different aspects.

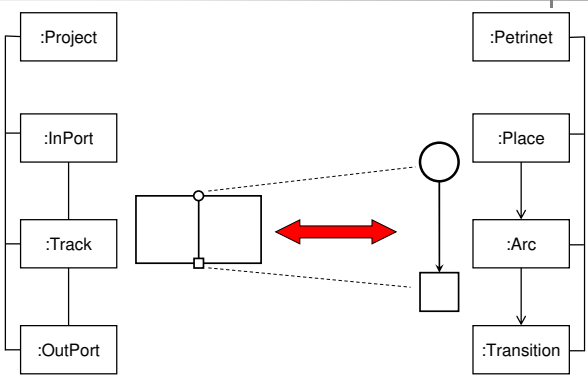
Defining Transformations

Example borrowed from ComponentTools

engineer
„practice“

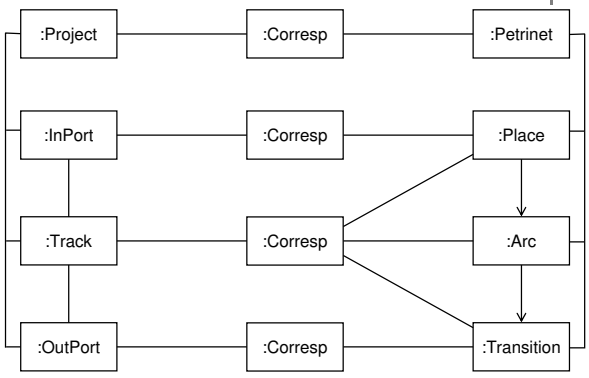
formal methods
„theory“

Transformation Rule (Idea)



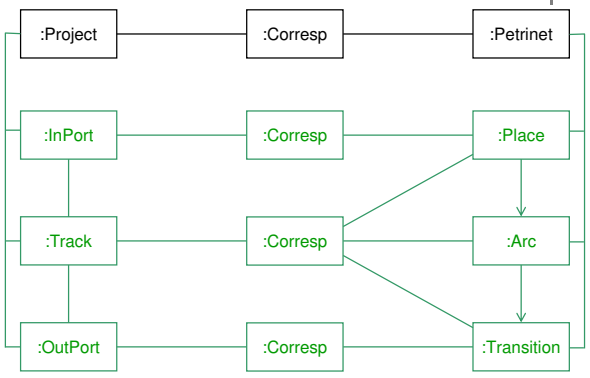
Transformation Rule (Idea)

DTU IMM E. Kindler



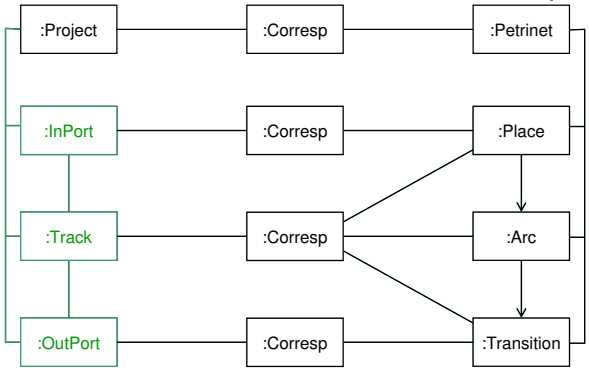
Triple Graph Grammar Rule (TGG)

DTU IMM E. Kindler



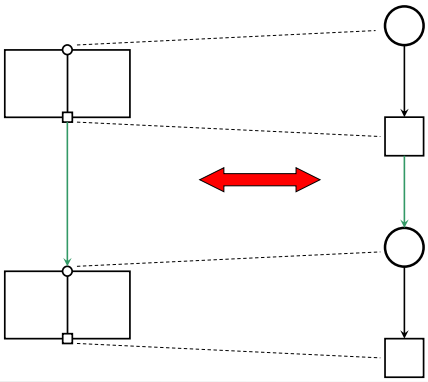
TGG-Rule Application

DTU IMM E. Kindler



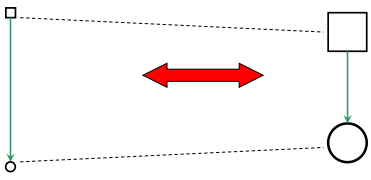
Transformation of two tracks

DTU IMM E. Kindler



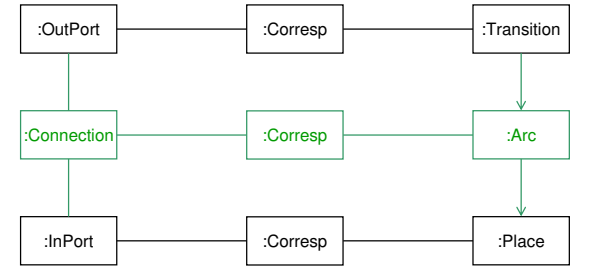
Transformation of two tracks

DTU IMM E. Kindler

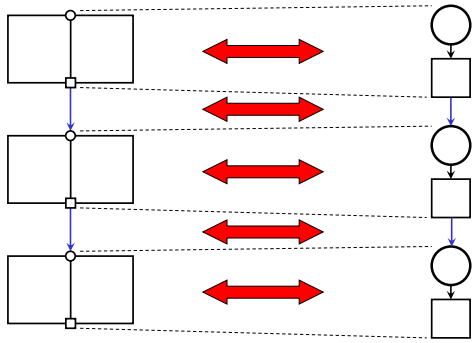


Transformationrule: Connection

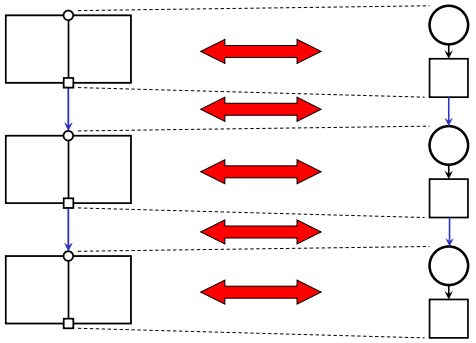
DTU IMM E. Kindler



Semantics of TGG-Rules



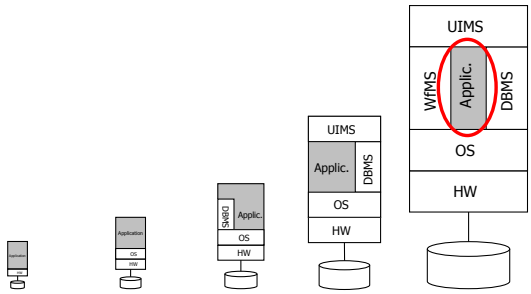
"Model-driven Execution"



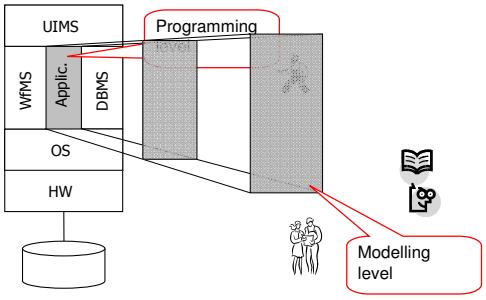
Modelling Behaviour

- In standard notation (story pattern)
- Transformations (TGGs)
- Definition and integration of complex behaviour AoM (automata and their synchronisation)

A historical view (v.d.Aalst 96)



History Continued



Theses

- We will always have programming and programmers!
- We should always teach programming!
- But, software engineers should be trained in their engineering and modelling skills!
- And this is where they should be at their best!
- Most of the rest can be automated!
- Eventually, programming will be for software engineers as assembler is today for programmers.