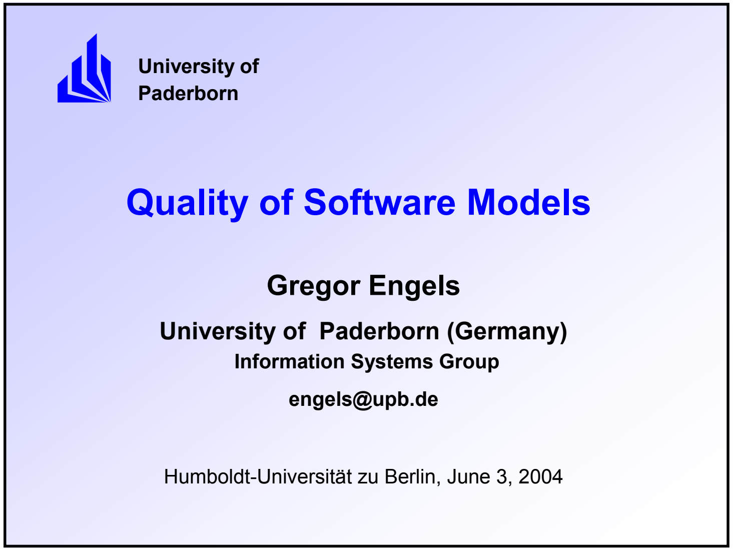


University of
Paderborn

Quality of Software Models

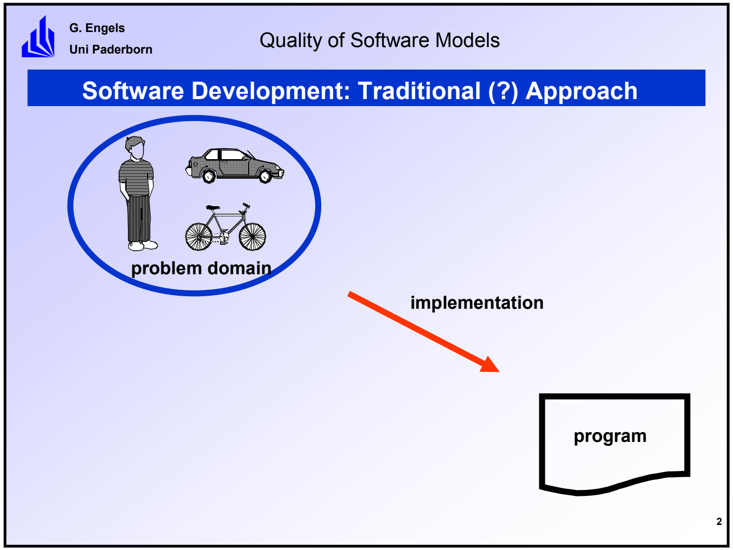
Gregor Engels

University of Paderborn (Germany)

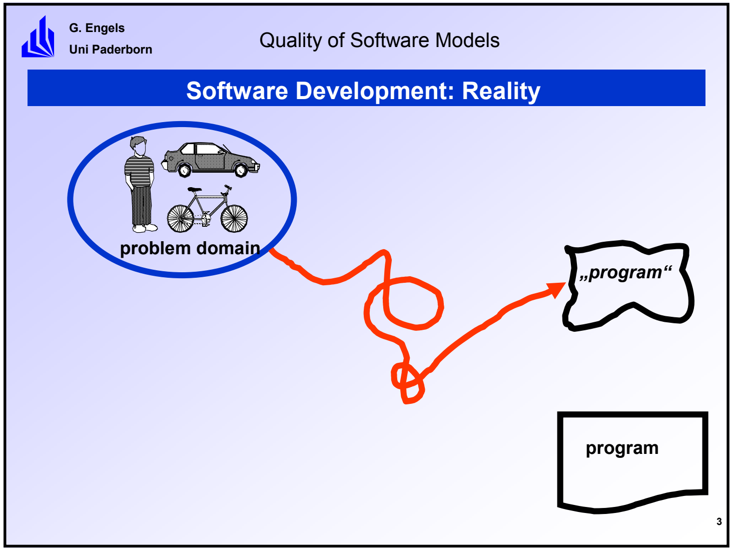
Information Systems Group

engels@upb.de

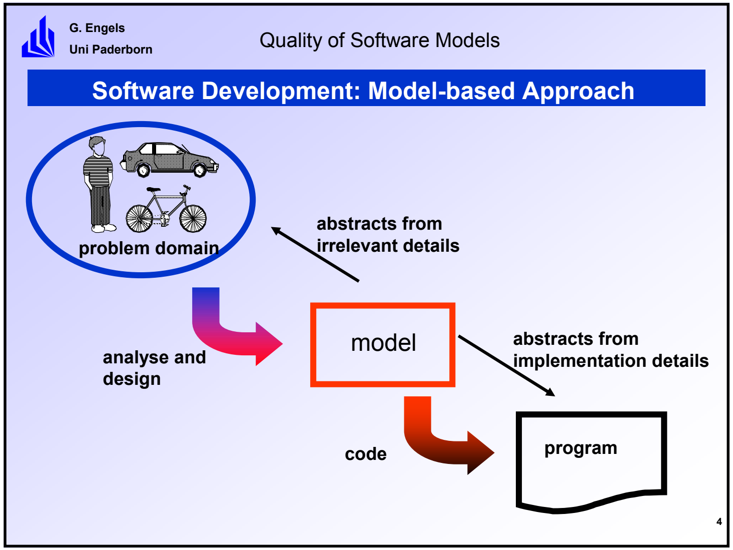
Humboldt-Universität zu Berlin, June 3, 2004



Software Development: Traditional (?) Approach

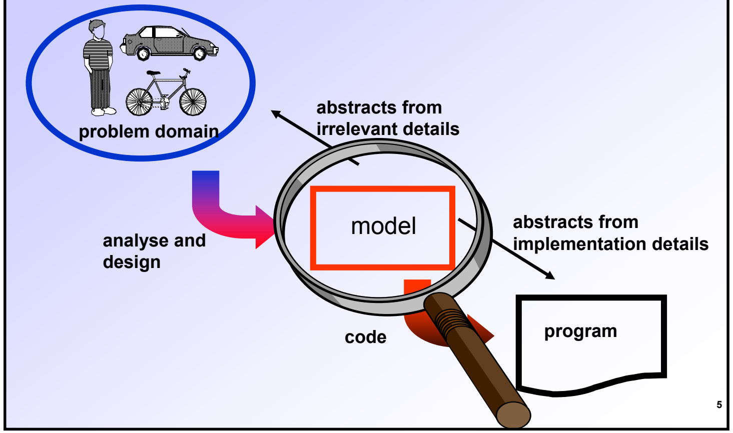


Software Development: Reality



Software Development: Model-based Approach

Software Development: Model-based Approach



problem domain

abstracts from irrelevant details

analyse and design

model

abstracts from implementation details

code

program

Outline of the Talk

1. Motivation
2. Characterization of Model Qualities
3. Model Quality Assurance Techniques
   - Generic Model Construction Approaches
   - Domain-specific Languages
   - Pattern-based Modelling
   - Model Analysis
4. Summary

2. Characterization of Model Qualities

review of typical software (= program) qualities

- correctness
- reliability
- robustness
- user friendliness
- understandability
- maintainability
- reusability
- portability
- interoperability

are there
also usable as
model qualities?

Model Qualities: Correctness

Correctness of programs:

- “a program is correct if it behaves according to a specification (= model)”

Correctness of models:

- “a model is correct if it reflects the user’s requirements” or
- “a model is correct if it fulfills additionally given properties”

Model Qualities: Reliability

Reliability of programs:

- “a program is reliable if the user can depend on it”

Reliability of models:

- “a model is reliable if the user (modeller, client, software developer) can depend on it”
- this means: model errors are not serious
  - are easily detected during coding
  - are obvious
  - model “similar“ situations

Model Qualities: Robustness

Robustness of programs:

- “a program is robust if it behaves reasonably even in unexpected, not specified situations”

Robustness of models:

- “a model is robust if unexpected, not specified situations do not have an impact on the model”
- this means that the model is “complete” w.r.t. all relevant requirements

Model Qualities: User Friendliness

User friendliness of programs:

- external quality
- “a program is user friendly if its human users find it easy to use.”

User friendliness of models:

- “a model is user friendly if its human users (modeller, client, software developer) find it easy to use (i.e. read or understand).”

Model Qualities: Understandability

Understandability of programs:

- internal quality
- “a program is understandable if programmers find it easy to understand.”

Understandability of models:

- “a model is understandable if its human users (modeller, client, software developer) find it easy to understand”

Note: User friendliness and understandability are the same for models, or?!

Model Qualities: Maintainability

Maintainability of programs:

- “a program is maintainable if corrective, adaptive, or perfective modifications can easily be made.”

Maintainability of models:

“a model is maintainable if corrective, adaptive, or perfective modifications can easily be made.”

Model Qualities: Reusability

Reusability of programs:

- “a program (component) is reusable if it can easily be integrated into a new software system.”

Reusability of models:

“a model (component) is reusable if it can easily be integrated into a new model.”

Model Qualities: Portability

Portability of programs:

- “a program is portable if it can run in different environments.”

Portability of models:

- “a model is portable if it can be read and modified by different CASE tools.”

- this means that the model can be exported and imported in a standardized exchange format (e.g. XML).

Model Qualities: Interoperability

Interoperability of programs:

- “a program is interoperable if it can coexist and cooperate with other programs.”

Interoperability of models:

- “a model is interoperable if it can coexist and cooperate with other models.”

- this means that horizontal consistency with other models is defined and given.

2. Characterization of Model Qualities

typical model qualities

- correctness
- reliability
- robustness
- user friendliness
- understandability
- maintainability
- reusability
- portability
- interoperability

are
program qualities
also usable as
model qualities?

yes – but they have
to be redefined or
re-interpreted

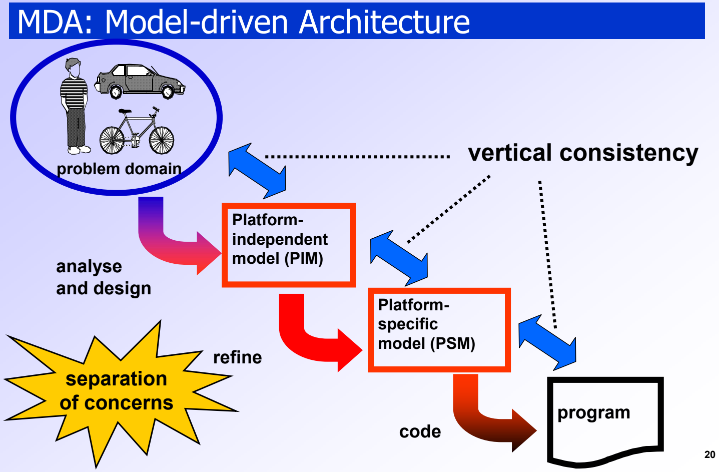
Outline of the Talk

1. Motivation
2. Characterization of Model Qualities
3. ➡ Model Quality Assurance Techniques
   - Generic Model Construction Approaches
   - Domain-specific Languages
   - Pattern-based Modelling
   - Model Analysis
4. Summary

Outline of the Talk

1. Motivation
2. Characterization of Model Qualities
3. Model Quality Assurance Techniques
   - ➡ Generic Model Construction Approaches
   - Domain-specific Languages
   - Pattern-based Modelling
   - Model Analysis
4. Summary

MDA: Model-driven Architecture



problem domain

vertical consistency

analyse and design

Platform-independent model (PIM)

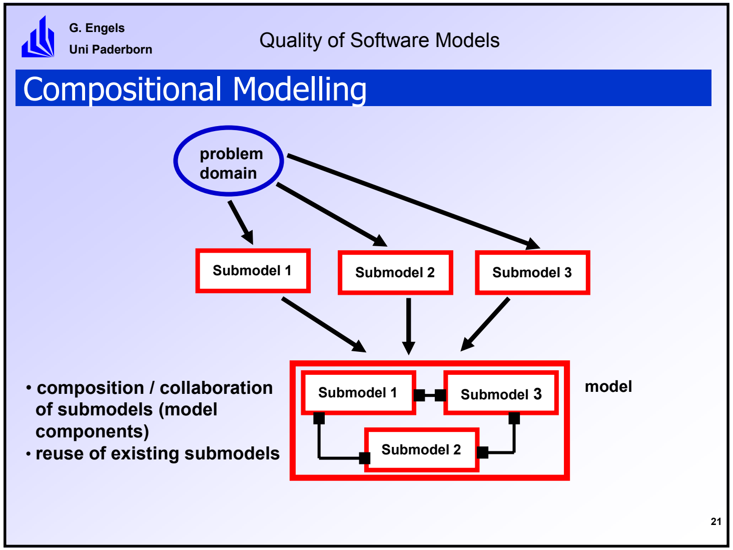
refine

Platform-specific model (PSM)

code

program

separation of concerns



Compositional Modelling

problem domain

Submodel 1

Submodel 2

Submodel 3

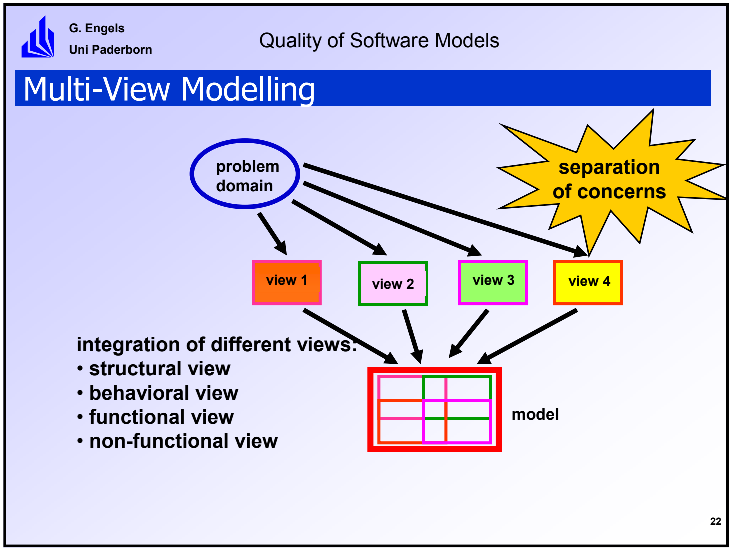
- composition / collaboration of submodels (model components)
- reuse of existing submodels

Submodel 1

Submodel 3

Submodel 2

model



Multi-View Modelling

problem domain

separation of concerns

view 1

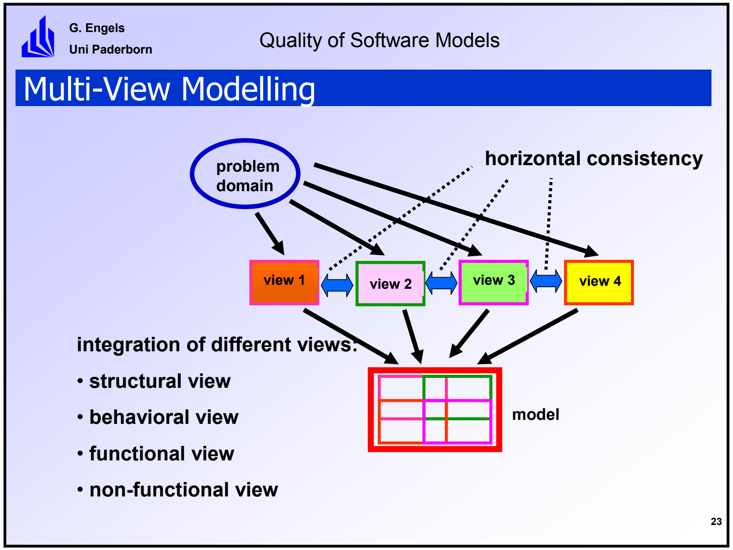
view 2

view 3

view 4

integration of different views:
- structural view
- behavioral view
- functional view
- non-functional view

model



Multi-View Modelling

problem domain

horizontal consistency

view 1

view 2

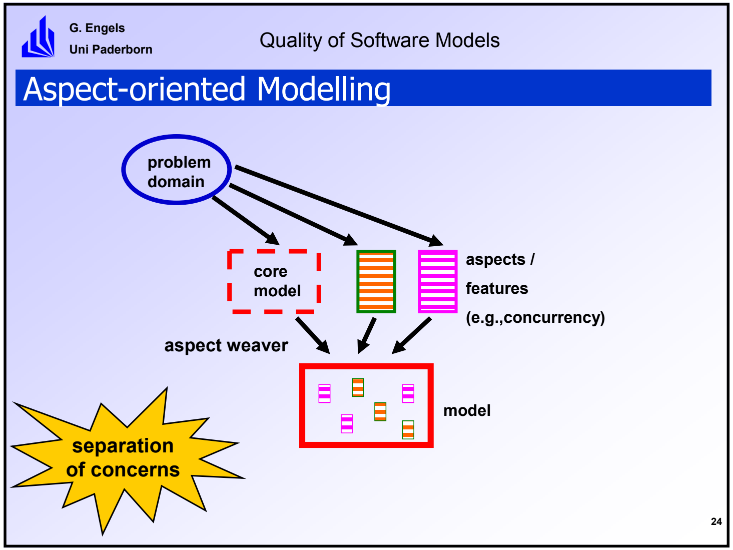
view 3

view 4

integration of different views:

- structural view
- behavioral view
- functional view
- non-functional view

model



Aspect-oriented Modelling

problem domain

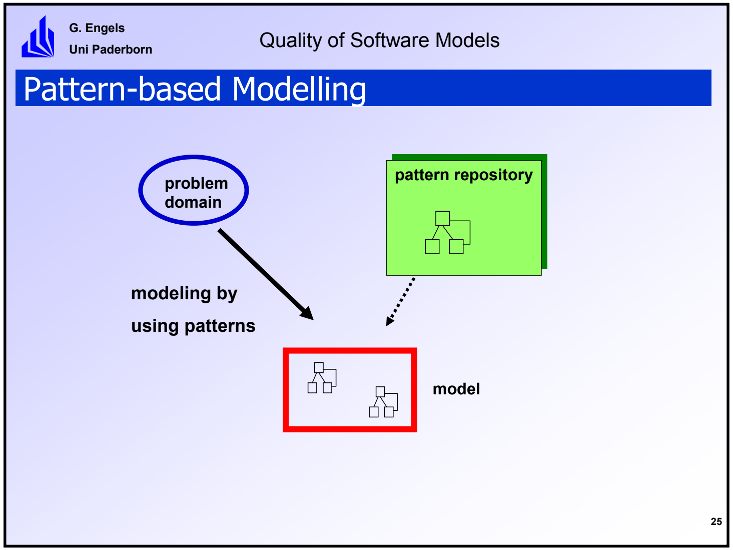
core model

aspects / features (e.g.,concurrency)

aspect weaver

model

separation of concerns



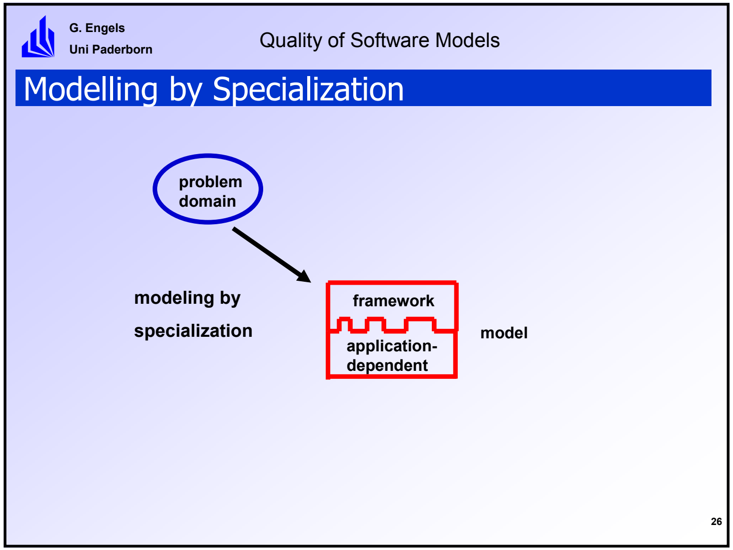
Pattern-based Modelling

problem domain

pattern repository

modeling by using patterns

model



Modelling by Specialization

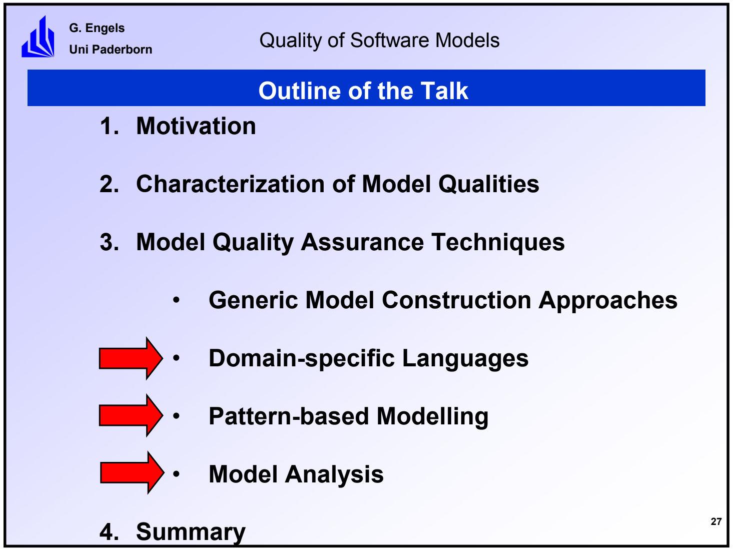
problem domain

modeling by specialization

framework

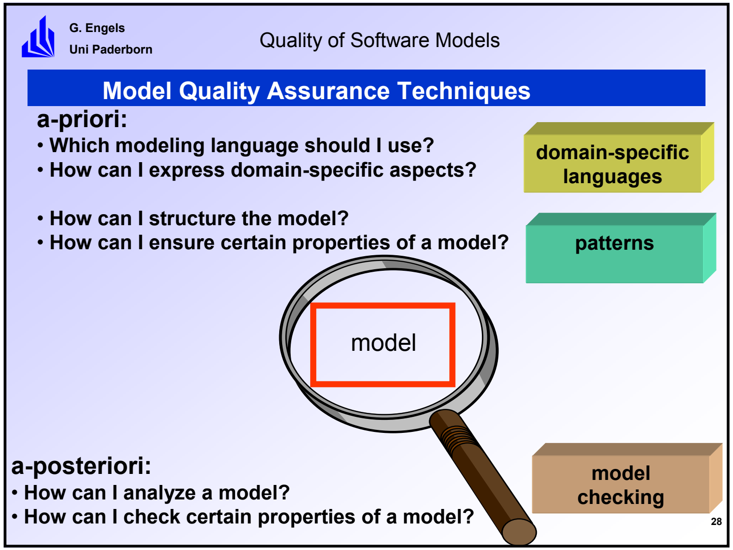
application-dependent

model



Outline of the Talk

1. Motivation
2. Characterization of Model Qualities
3. Model Quality Assurance Techniques
   - Generic Model Construction Approaches
   - ➡ Domain-specific Languages
   - ➡ Pattern-based Modelling
   - ➡ Model Analysis
4. Summary



Model Quality Assurance Techniques

a-priori:

- Which modeling language should I use?
- How can I express domain-specific aspects?

domain-specific languages

- How can I structure the model?
- How can I ensure certain properties of a model?

patterns

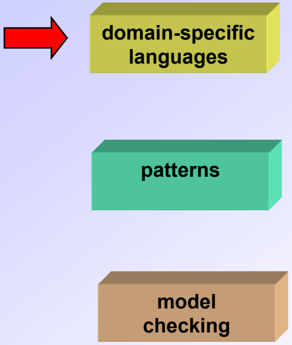
model

a-posteriori:

- How can I analyze a model?
- How can I check certain properties of a model?

model checking

Model Quality Assurance Techniques



Stefan Sauer

Alexander Förster

Jochen Küster

The Beauty of High-Tech Dashboards

© CAAAG

Infotainment System Features

- entertainment (radio, CD, MP3)
- convenience (GPS, navigation)
- security (feature access)
- information services (traffic, weather, calendar, …)
- Internet access (synch with service providers)
- user interface consolidation (with vehicle control)
- service integration (in case of accidents, problems)
- communication (docking stations for cell phones, PDA)
- commerce (toll collect!)

Driver Safety

Driver Safety

Danger

- **information overload**
- **driver distraction**

Some existing approaches

- **multi modal, multi media interfaces**
- **prioritizing information presented to driver**
- **integrated system approach**
  (e.g. no new info while braking)
- **context-sensitive options**
  (e.g. speedometer only while car in motion)

Software Engineering Challenges

Research Questions (1):

- **integrated design of user interface and functionality**
- **component integration**
  - **user interface, functionality**
  - **consistency**
  - **coordination**
  - **prioritizing / context-sensitive adaption**
- **component configuration / update / exchange**
- **end-user adaption**

Software Engineering Challenges

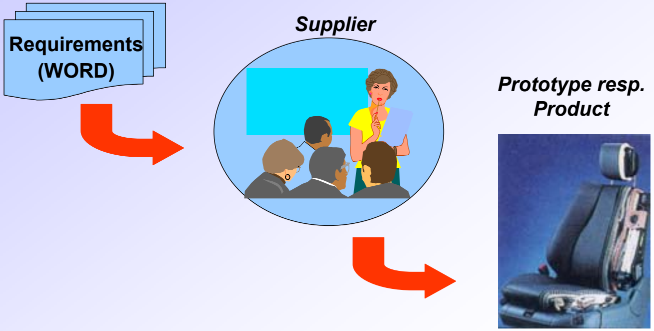
Research Questions (2):

- **appropriate modeling concepts**
- **appropriate modeling language**
- **appropriate development process**
  - **role of**
    - **software engineers**
    - **electrical engineers**
    - **designers**
    - **suppliers**
    - **…**

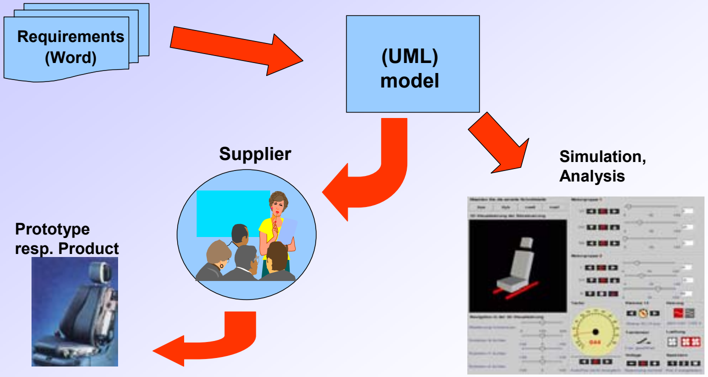
Traditional Development Process

75 % of car production costs are supplier costs!

OEM



Model-based Development



Requirements (Word)

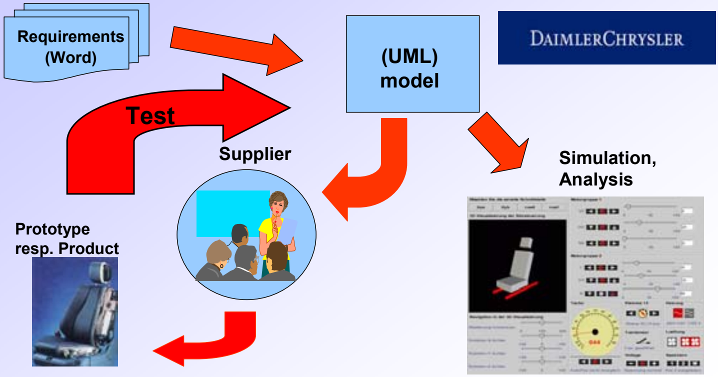
(UML) model

Supplier

Simulation, Analysis

Prototype resp. Product

Model as Test Specification



Requirements (Word)

(UML) model

DAIMLERCHRYSLER

Test

Supplier

Simulation, Analysis

Prototype resp. Product

Focus of our Research

Research Questions (1):

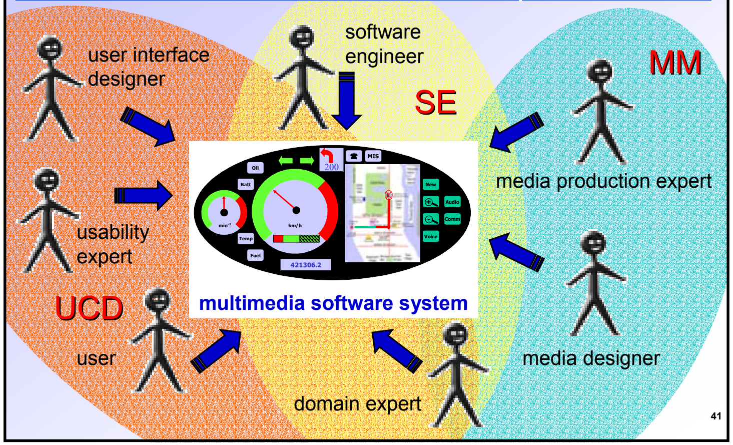
- integrated design of user interface and functionality
- component integration
  - user interface, functionality
  - consistency
  - coordination
  - prioritizing / context-sensitive adaption
- component configuration / update / exchange
- end-user adaption

Focus of our Research

Research Questions (2):

- appropriate modeling concepts
- appropriate modeling language
- appropriate development process
  - role of
    - software engineers
    - electrical engineers
    - designers
    - suppliers
    - …

Multimedia Software Development



user interface designer

software engineer

SE

MM

media production expert

usability expert

UCD

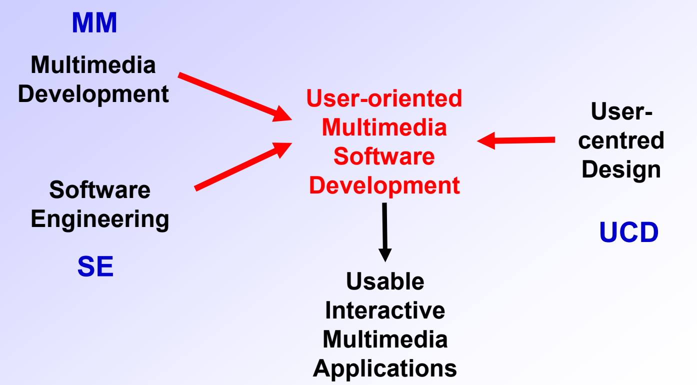
user

multimedia software system

media designer

domain expert

Objective: What we want!



MM

Multimedia Development

Software Engineering

SE

User-oriented Multimedia Software Development

User-centred Design

UCD

Usable Interactive Multimedia Applications

MM & SE =MMSE: What we have!

OMMMA =

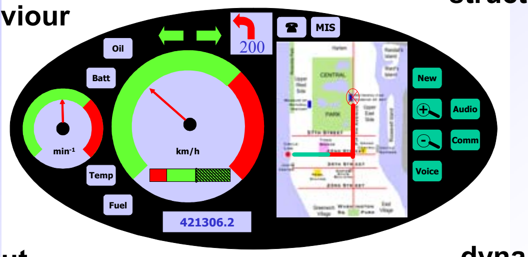
Object-oriented Modeling of MultiMedia Applications

- **model-based, object-oriented multimedia software engineering**
- **method**
- **language OMMMA-L defined as UML profile**
- **supports integrated modeling of different aspects of interactive multimedia applications**

Aspects of an OMMMA Model

interactive behaviour

structure

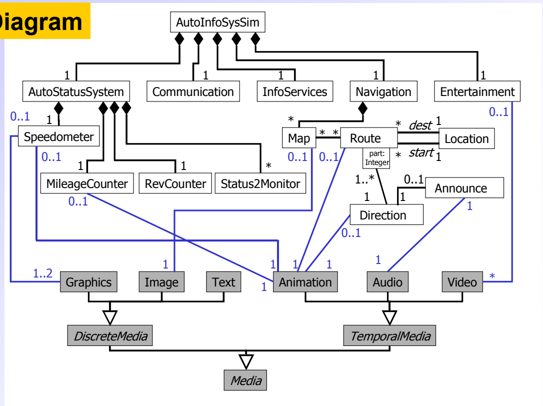


layout

dynamic behaviour

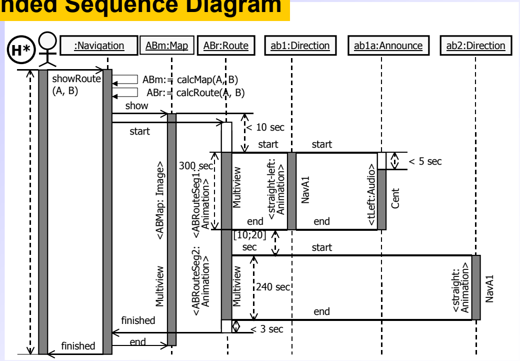
OMMMA-L: the Structure Aspect

Class Diagram



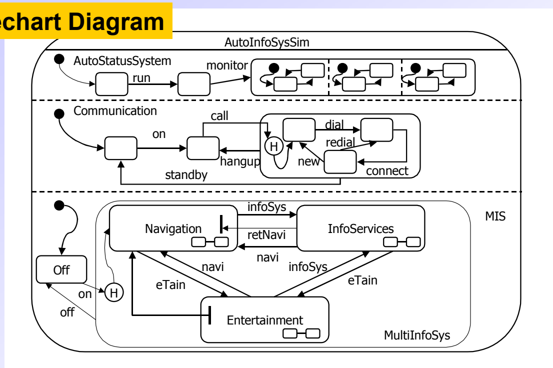
OMMMA-L: the Dynamic Behaviour Aspect

Extended Sequence Diagram



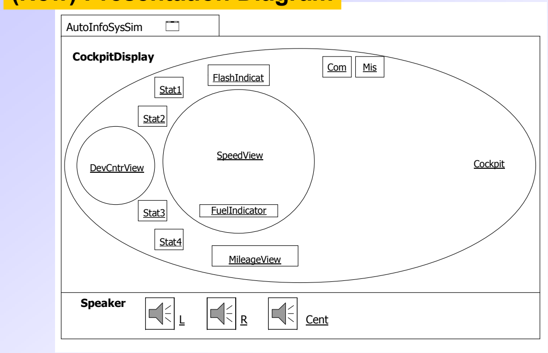
OMMMA-L: the Interactive Behaviour Aspect

Statechart Diagram



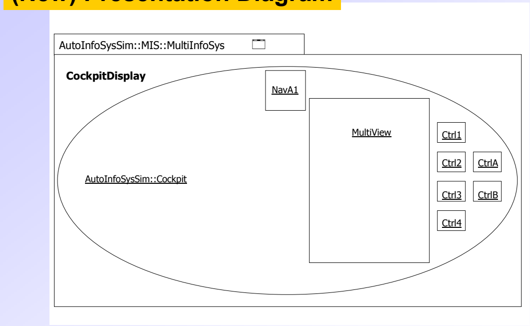
OMMMA-L: the Layout Aspect

(New) Presentation Diagram

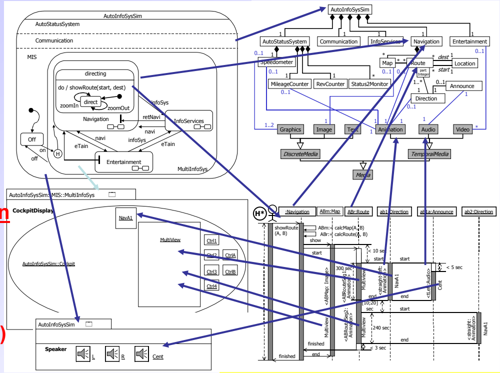


OMMMA-L: the Layout Aspect

(New) Presentation Diagram



OMMMA: 4 Fundamental Views



statechart: interactive control

presentation diagram: spatial layout, UI structure (not in UML)

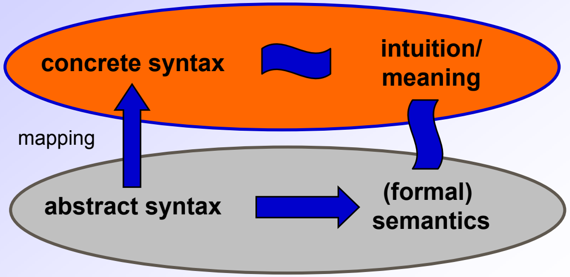
class diagram: media and application structure

sequence diagram: temporal behavior

syntactic consistency constraints

Language Engineering

(visual) [modeling] language



concrete syntax ~ intuition/ meaning

end-user aspect

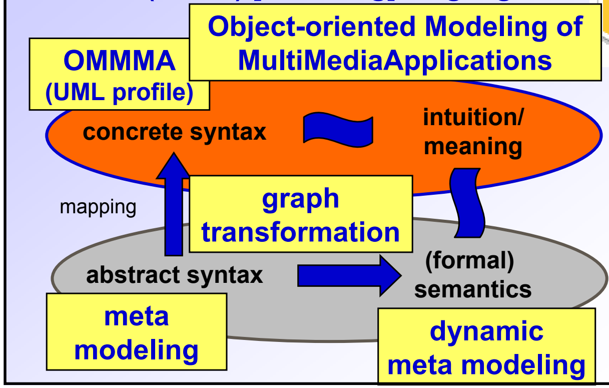
mapping

abstract syntax → (formal) semantics

language definition aspect

Language Engineering

(visual) [modeling] language



OMMMA (UML profile)

Object-oriented Modeling of MultiMediaApplications

concrete syntax ~ intuition/ meaning

end-user aspect

mapping

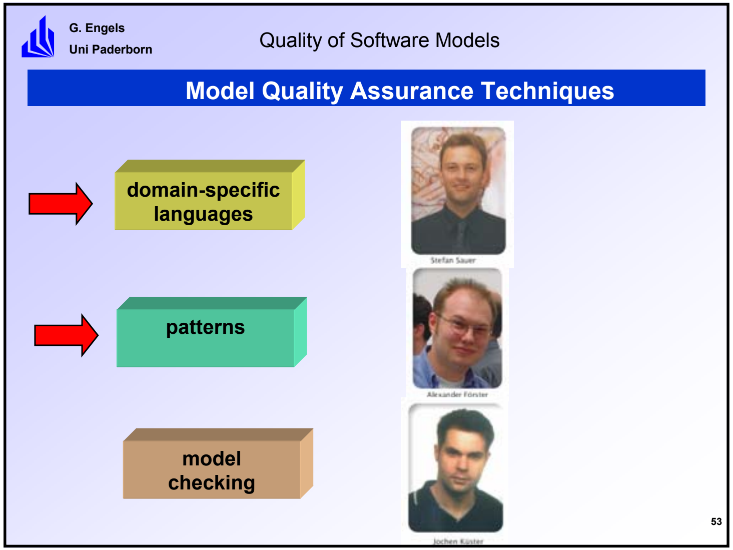
graph transformation

abstract syntax → (formal) semantics

meta modeling

dynamic meta modeling

language definition aspect



Model Quality Assurance Techniques

domain-specific languages

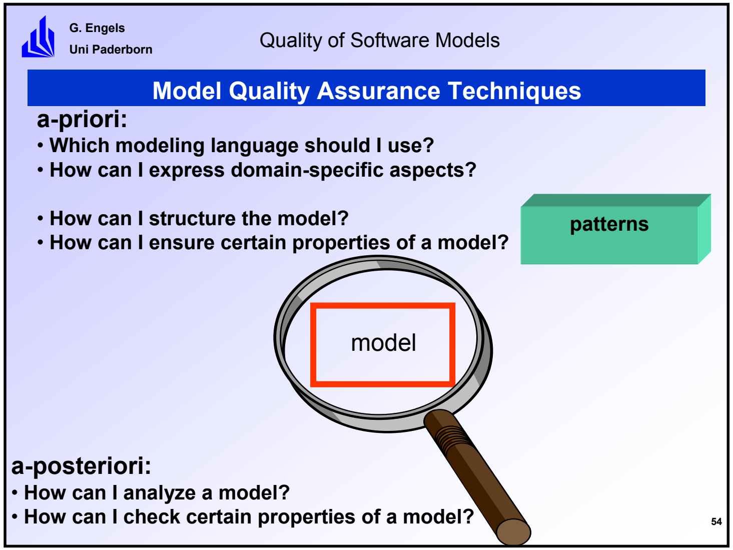
patterns

model checking

Stefan Sauer

Alexander Förster

Jochen Küster



Model Quality Assurance Techniques

a-priori:

- Which modeling language should I use?
- How can I express domain-specific aspects?

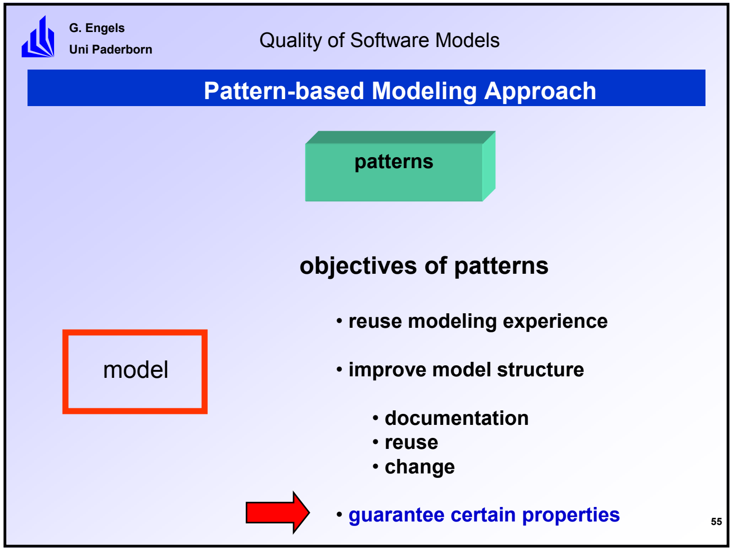
- How can I structure the model?
- How can I ensure certain properties of a model?

patterns

model

a-posteriori:

- How can I analyze a model?
- How can I check certain properties of a model?



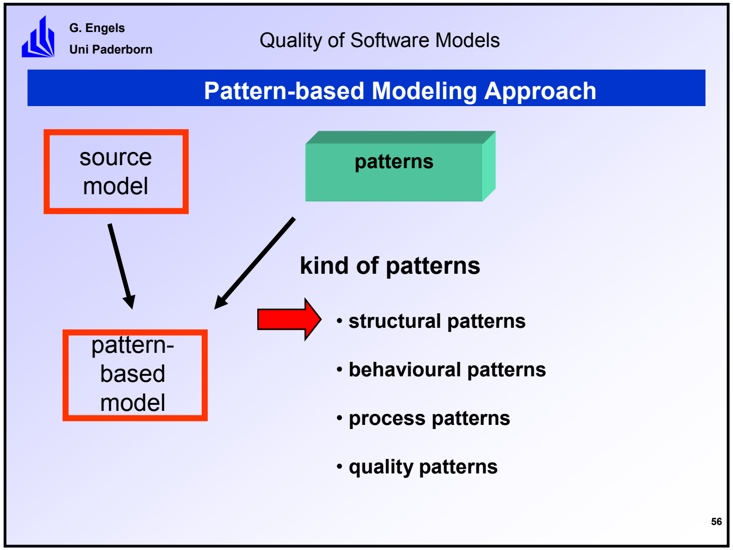
Pattern-based Modeling Approach

patterns

model

objectives of patterns

- reuse modeling experience
- improve model structure
  - documentation
  - reuse
  - change
- guarantee certain properties



Pattern-based Modeling Approach

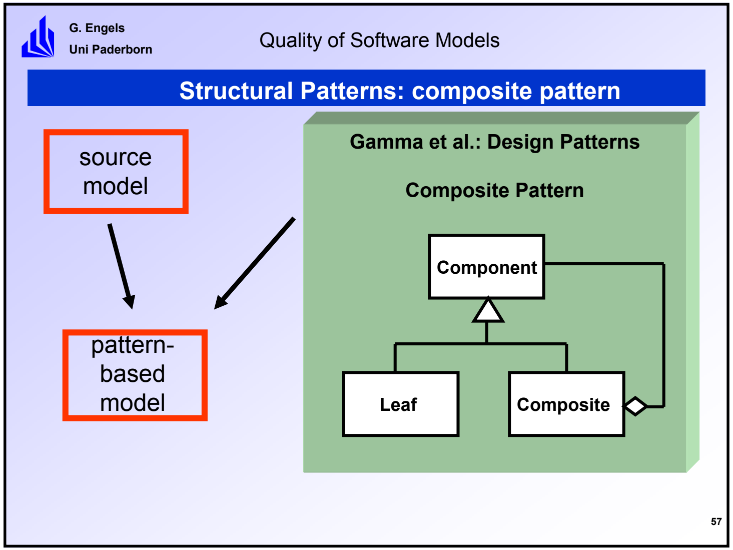
source model

patterns

pattern-based model

kind of patterns

- structural patterns
- behavioural patterns
- process patterns
- quality patterns



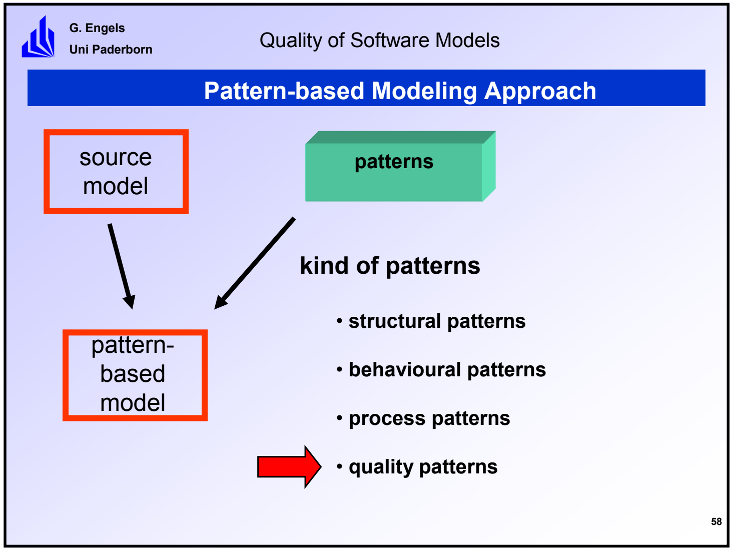
Structural Patterns: composite pattern

source model

pattern-based model

Gamma et al.: Design Patterns

Composite Pattern



Pattern-based Modeling Approach

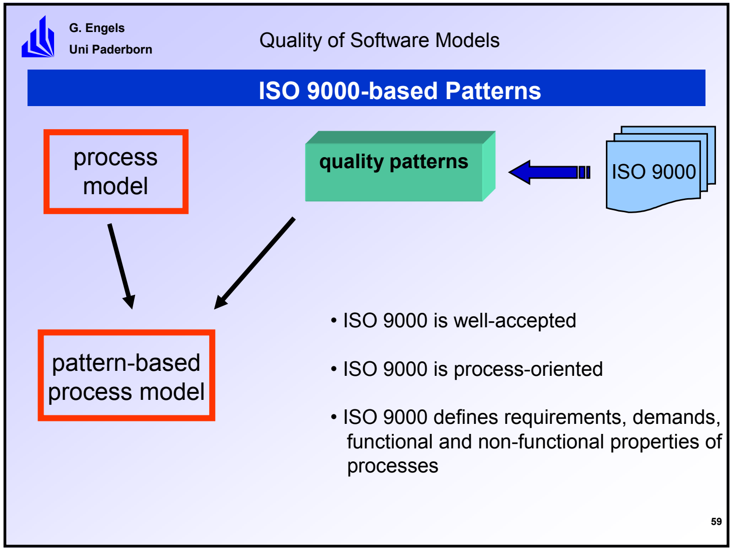
source model

patterns

pattern-based model

kind of patterns

- structural patterns
- behavioural patterns
- process patterns
- quality patterns



ISO 9000-based Patterns

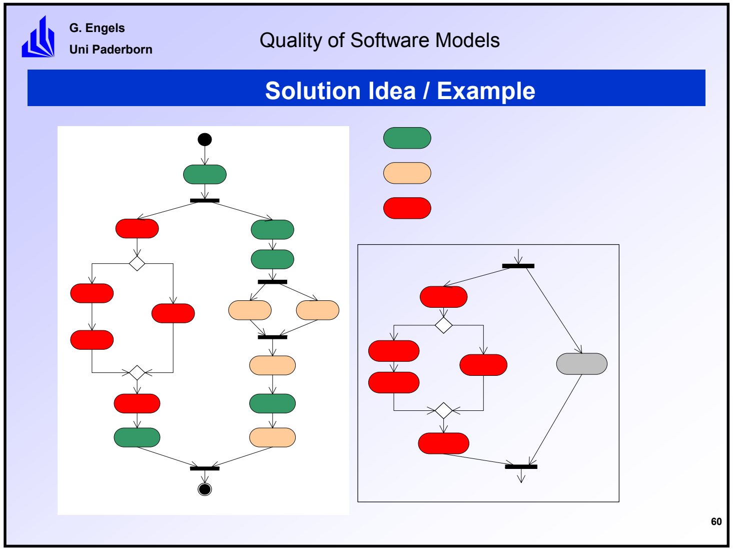
process model

quality patterns

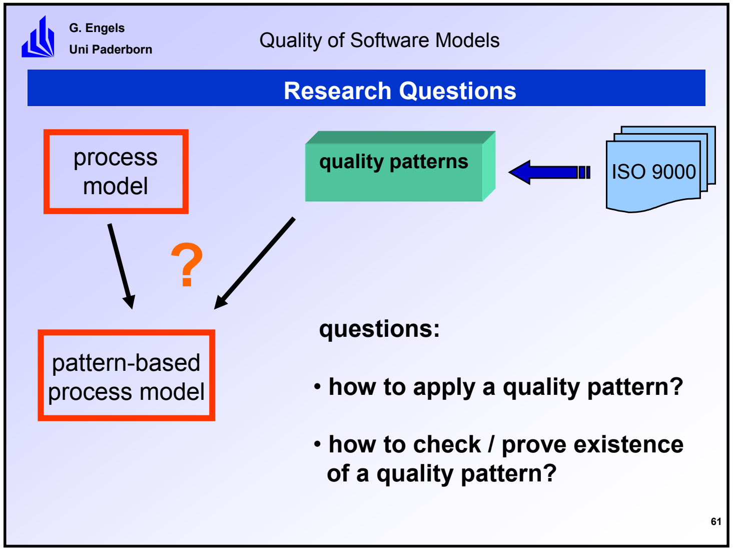
ISO 9000

pattern-based process model

- ISO 9000 is well-accepted
- ISO 9000 is process-oriented
- ISO 9000 defines requirements, demands, functional and non-functional properties of processes



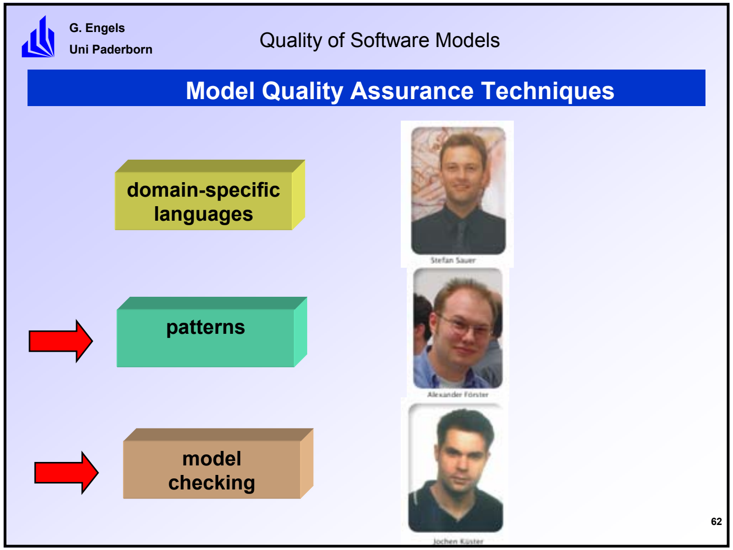
Solution Idea / Example



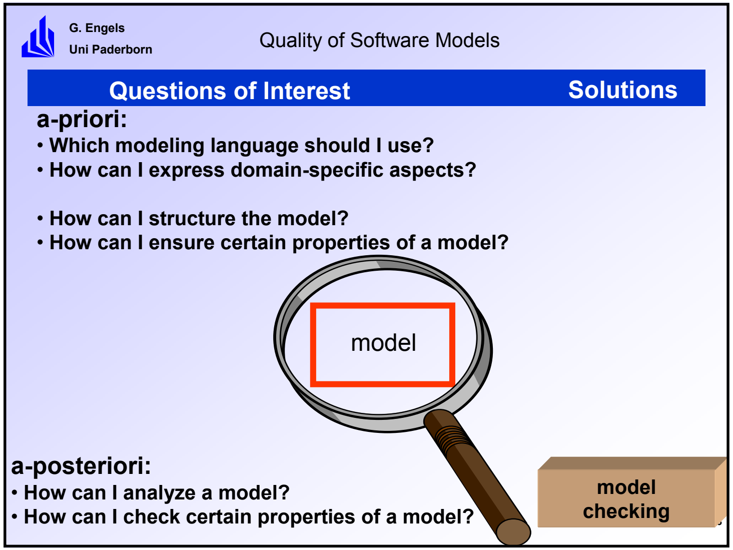
Research Questions

questions:

- how to apply a quality pattern?
- how to check / prove existence of a quality pattern?



Model Quality Assurance Techniques



Questions of Interest Solutions

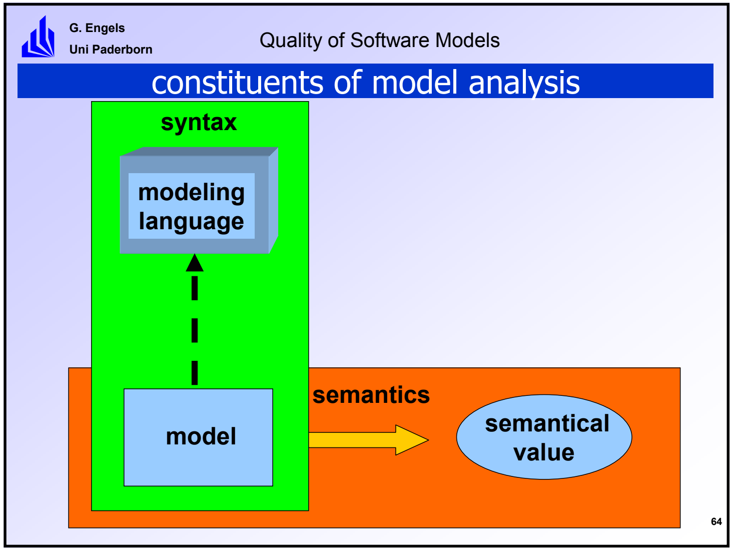
a-priori:

- Which modeling language should I use?
- How can I express domain-specific aspects?

- How can I structure the model?
- How can I ensure certain properties of a model?

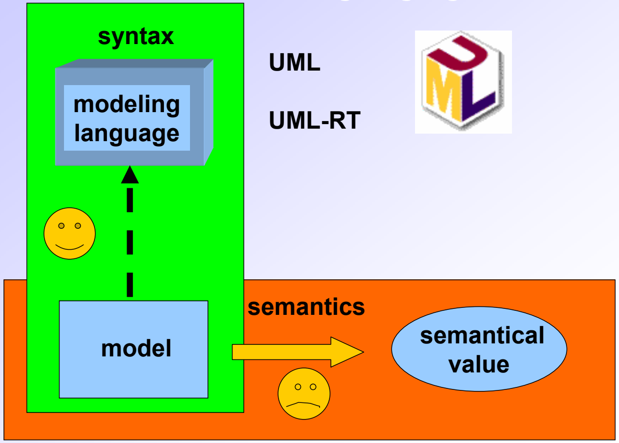
a-posteriori:

- How can I analyze a model?
- How can I check certain properties of a model?



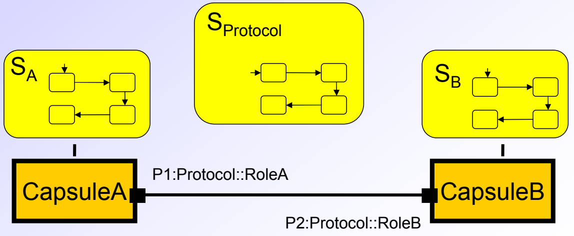
constituents of model analysis

choice of modeling language



Concepts of UML-RT

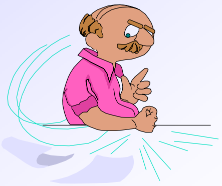
- **UML-RT** is a UML profile
- introduces the concepts of **capsule**, port, protocol, protocol role, connector (as stereotypes of existing UML constructs)
- capsule collaboration diagram is a special form of a collaboration
- associates a specific interpretation to **capsule statecharts** and **protocol statecharts**



semantics of modeling language

Problems

- **multi-paradigms**
- **multi-views**
- **application domain-dependent**
- **development phase-dependent**
- **theoreticians have no answer (yet)**



semantics of modeling language

software engineer

Our solution:

define for each consistency property a partial, i.e. consistency property-dependent semantics

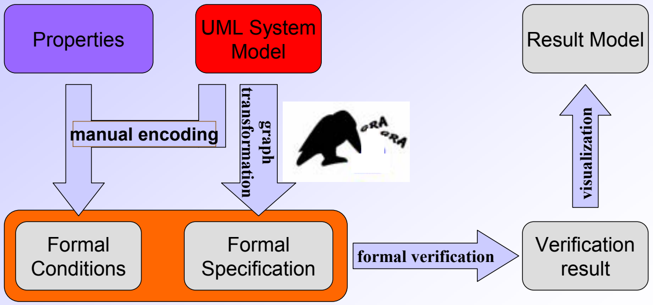
Consistency Management Process

- **Step 1:** identification of the consistency problem
- **Step 2**: choice of an appropriate semantic domain, i.e., where consistency constraint can be checked
- **Step 3:** definition of partial semantical mapping
- **Step 4:** formal specification of consistency constraints
- **Step 5:** tool-based verification of consistency constraints

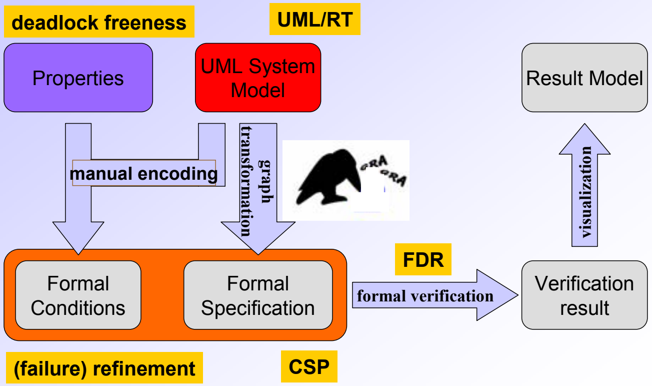
Instantiated Consistency Management Process

- **Step 1:** identification of the consistency problem
  property ≡ „deadlock-freeness“
- **Step 2**: choice of an appropriate semantic domain, i.e., where consistency constraint can be checked
  Communicating Sequential Processes (CSP)
- **Step 3:** definition of partial semantical mapping
  Mapping of capsule and protocol statecharts to CSP
- **Step 4:** formal specification of consistency constraints
  Trace / failure refinement of processes in CSP
- **Step 5:** tool-based verification of consistency constraints
  Use of FDR model checker

Consistency: Model-based Verification of Properties



Consistency: Model-based Verification of Properties



Outline of the Talk

1. Motivation
2. Characterization of Model Qualities
3. Model Quality Assurance Techniques
   - Generic Model Construction Approaches
   - Domain-specific Languages
   - Pattern-based Modelling
   - Model Analysis
4. Summary

Impact of Quality Assurance Techniques on Model Qualities

	domain-specific languages	patterns	model checking
correctness			
reliability			
robustness			
user friendliness			
understandability			
maintainability			
reusability			
portability			
interoperability			

Impact of Quality Assurance Techniques on Model Qualities

	domain-specific languages	patterns	model checking
correctness			☺
reliability	☺	☺	
robustness	☺		
user friendliness	☺	☺	
understandability	☺	☺	
maintainability	☺	☺	
reusability			☺
portability			
interoperability			☺

Interest in related Research?

European project

Research and Training Network (RTN)

SegraVis
Syntactic and Semantic Integration of Visual Modelling Techniques

Syntactic and Semantic Integration of Visual Modeling Techniques

2002 - 2006

SegraVis - Grants

SegraVis
Syntactic and Semantic Integration of Visual Modelling Techniques

Grants are available for pre- and post-doc‘s!

- arguments
  - flexible timing and organization
  - only a few bureaucracy
  - local contracts / local rules
  - no further application in Brussels

www.segravis.org

Contact one of the partners!

The End