Joint work with …

G. Engels, R. Heckel, J. Küster:

J. Küster, R. Heckel, G. Engels:

G. Engels, J. Küster, L.P.G. Groenewegen:

available at http://www.upb.de/cs/ag-engels
History of Software Development

• from small programs

• to large programs
History of Software Development

• from small programs
• to large programs
• to software systems

History of Software Development

• from small programs
• to large programs
• to software systems
• to layered systems
History of Software Development

- from small programs
- to large programs
- to software systems
- to layered systems
- to networks of components

networks of components

- components fulfill a certain functionality
- components interact
  - send messages
  - wait for / receive messages
networks of components

- networks may evolve statically
- components are replaced

- networks may evolve dynamically
- ad-hoc networks
- mobile applications
networks of components

Question: „does the new component fit?“

What does this mean?
• „consistent interaction between components“
• „new component consistent with requirements“

networks of components

Question: „does the new component fit?“

What does this mean?
• „consistent interaction between components“
• „new component consistent with requirements“
Observation:

• consistency is an often used notion!

• but, consistency is also a very unclear notion!

Research Questions:

• what is meant by consistency?

• how to check consistency?

consistent interaction of components?

component

Research Questions:

• what is meant by consistency?

• how to check consistency?
consistent interaction of components?

Research Questions:
• what is meant by consistency?
• how to check consistency?

How to check consistency?

basic idea:
• consistency has to be checked on model level
• keep model information with component code
• „model-carrying code“
Model-Centered Software Development

Problem domain

Analyse and design

Model

Component

Component

Component

Code

Model-carrying components
Model-Centered Software Development

Consequence:
Consistency checking can be lifted to the model level!

Consistency Management

Refined research questions:

- what is meant by consistency of models?
- how to check consistency of models?
Working Definition

A model $m$ is consistent with respect to a property $p$ iff $m$ fulfills the property $p$.

**syntactical consistency**

property $p = "\text{syntactical correctness}\"$

$m$ fulfills the property $p = "\text{m is syntactically correct}\"$
Working Definition

A model $m$ is consistent with respect to a property $p$ iff $m$ fulfills the property $p$.

Semantical consistency: example 1

Property $p_1 =$ „there are no contradictions“

$m$ fulfills the property $p_1 =$ „$m$ can be mapped to a value in the semantical domain“

Semantical consistency: example 2

Property $p_2 =$ „deadlock-freedom“

$m$ fulfills the property $p_2 =$ „semantics of $m$ is deadlock-free“
Working Definition

A model \( m \) is consistent with respect to a property \( p \) iff \( m \) fulfills the property \( p \).

Semantical consistency: example 3

Property \( p_3 = \) „constraints given by a model \( r_m \)“

\( m \) fulfills the property \( p_3 = \) „\( m \) is a refinement of \( r_m \)“

Constituents of consistency checking

Syntax

Modeling language

Model

Semantics

Semantical value
choice of modeling language

- **syntax**
  - UML
  - UML-RT

- **model**

- **semantics**
  - semantical value

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
choice of modeling language

- syntax
- UML-RT
- modeling language
- model
- semantics
- semantical value

semantics of modeling language

Problems
- multi-paradigms
- multi-views
- application domain-dependent
- development phase-dependent
- theoreticians have no answer (yet)
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 p2 = "deadlock-freedom"
  property p3 = "constraints given by a model r_m"

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

Step 1: Horizontal Consistency of Deadlock-Freedom

- Intuitive Semantics:
  - messages sent by a capsule are transmitted via the connector to the other capsule
  - connector has a certain capacity and latency
**Step 1: Horizontal Consistency of Deadlock-Freedom**

- **Consistency**
  - **Overlap:**
    - Communication and interaction aspect
    - Between send and receive operations of a message in different statecharts
  - **Desired Semantic Consistency:**
    - Statechart behavior must be compatible (e.g., each receiving operation must have an earlier sending operation of that message)

**Step 2: CSP as a Semantic Domain**

- Formal notation for concurrent systems
- Collection of mathematical models
- Syntax of CSP-processes:
  \[ P ::= \text{STOP} \mid a \rightarrow P \mid P ? P \mid P \cdot P \mid P \setminus a \mid pn \]
- Semantic model: traces and failures
- Trace is a finite sequence of actions
- Failure \((s, A)\) is a trace \(s\) and a set of actions \(A\) refused after \(s\)
- Trace refinement: \(P \subseteq T Q \iff \text{traces}(Q) \supseteq \text{traces}(P)\)
- Failures refinement: \(P \subseteq F Q \iff \text{failures}(Q) \supseteq \text{failures}(P)\)

CSP (Communicating Sequential Processes)
- Provides a language for behavioral models
- Allows the description of relationship of models
- Tool support is available
Step 3: Mapping into the Semantic Domain

- Mapping of a statechart to a CSP representation of a statechart is possible

Slide 38

Compound rule:
- Consists of UML pattern and CSP production \( L_T ::= R_T \)
- Special case: UML model is not transformed \( L_S = R_S \)
- Coupled by use of common variables
- Non-terminals that are replaced by following rules
The Concept of a Transformation Unit

A transformation unit (TU) consists of
• a set of compound transformation rules
• a control expression determining how to apply the rules

Control expression:
• p : apply rule p once
• p! : apply rule p as long as possible
• \{p_1, .., p_n\} : apply any of the rules p_1...p_n non-deterministicly
• \langle p_1, .., p_n \rangle : apply rules p_1...p_n in this order once
• combinations: \langle p1!,\{p2,p3\}\rangle

We have developed transformation units for
• Statecharts to CSP, UML collaborations to CSP (about 60 rules)
• Sequence Diagrams, Statecharts to Timed Automata

Step 3: Mapping into the Semantic Domain

• concentrate on aspects that are relevant to the consistency problem
• view process with respect to a port concentrates on those signals relevant to the interaction in focus
• in CSP this can be achieved by hiding events and actions not related to the port
Step 3: Mapping into the Semantic Domain

- model behavior of connector as a separate connector process

\[ V_{p1}(\text{CSP}(S_A)) \quad \text{CON} \quad V_{p2}(\text{CSP}(S_B)) \]

Step 4: Specification of Consistency Conditions

- compatibility of statechart behavior can be checked in CSP by requiring deadlock freedom

Consistency Condition 1 (Horizontal Consistency):
The process \( V_{p1}(\text{CSP}(S_A)) \parallel \text{CON} \parallel V_{p2}(\text{CSP}(S_B)) \) is deadlock free.
Step 5: Use of FDR tool support

UML-RT Model

Semantics of UML-RT (informal)

Semantic Domain: CSP

Consistency Condition

Model is consistent

or

Model is inconsistent

Step 1: identification of the consistency problem
property p2 = "deadlock-freeness"
property p3 = "constraints given by a model r_m"

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
Step 1: Vertical Consistency Problem in UML-RT

- Intuitive Semantics of protocol:
  - describes message exchange over the connector

- Overlap:
  - description of message exchange in the protocol
  - description of interaction induced by the capsule statecharts and intuitive connector behavior

- Desired Semantic Consistency:
  - conformance notion between the two descriptions
Step 4: Specification of Consistency

Consistency Condition 2 (Vertical Consistency):

\[ \text{CSP}(S_{\text{protocol}}) \sqsupseteq \text{V}_{p1}(\text{CSP}(S_A)) \bowtie \text{CON} \bowtie \text{V}_{p2}(\text{CSP}(S_B)) \]

- In CSP conformance can be established using refinement notion
- Failures refinement: interaction should exhibit complete protocol

Summary

- model-carrying components
- general methodology for specifying and analysing consistency
  - Step 1: identification of consistency problem
  - Step 2: choice of a semantic domain
  - Step 3: definition of partial mapping into the semantic domain
  - Step 4: specification of consistency conditions
  - Step 5: use tool support
- application of methodology to UML-RT
  - identification of consistency problems
  - methodology applied to two consistency problems
- characteristics of the methodology
  + not relying on the existence of a complete formal semantics
  + flexible with respect to consistency conditions
  + use of existing tool support
  - requires deep knowledge of various semantic domains
  - ...
Current / Future Work

- provide support for identification of consistency problems
- apply the methodology to other consistency problems
- provide guidelines how to choose the appropriate semantic domain
- realize a consistency workbench