PLAY-OUT FOR HIERARCHICAL COMPONENT ARCHITECTURES

Jörg Holtmann, Matthias Meyer
Automotive Software Engineering, 17. September 2013
Introduction and Motivation

Automotive Domain

- Approach to cope with complexity: component-based software development
Introduction and Motivation

Automotive Domain

Approach to cope with complexity: component-based software development

System Requirements Analysis

System Architecture Design

Software Requirements Analysis

Hardware Requirements Analysis

Software Design

Hardware Design

Software Integration Test

Software Construction

Module Test

Hardware Construction
Running Example
Body Control Module (BCM)

<table>
<thead>
<tr>
<th>ID</th>
<th>BCM System Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>49</td>
<td>When the remote key sends a central door locking request, the BCM has to send lock commands to all door locks.</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>236</td>
<td>When locking the doors was successful, the door locks have to confirm this by means of an acknowledgment message.</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>524</td>
<td>After the door locks were successfully locked, the BCM has to send a feedback command to the turn signal actuators.</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>851</td>
<td>When the speed threshold $\text{spd_thrsh}$ is exceeded, the body control module has to send commands to all door locks in order to lock the doors.</td>
</tr>
</tbody>
</table>
Introduction and Motivation
Formal Requirements and Hierarchical Component Architectures

Previous work

- Formal, scenario-based requirements engineering approach
- Not suited for hierarchical component architectures!

How to specify formal requirements for hierarchical component architectures across several abstraction levels?
CONTENTS

1. Introduction + motivation

2. Preliminaries: Previous approach on formal requirements engineering

3. Problem description:
   Approach not sufficient for hierarchical component architectures

4. Adapted modeling approach

5. Simulative validation of requirements for hierarchical component architectures
Previous work
Formal Requirements Engineering with Modal Sequence Diagrams (MSDs)

- Modal Sequence Diagrams (MSDs)
  - UML-compliant variant of Live Sequence Charts
  - Formal and modal semantics

Environment elements

System under development

msd RemoteLockDoors

rk: RemoteKey

bcm: Body ControlModule

dl_driver: DoorLock
dl_passenger: DoorLock

lockCenReq

c = 0

c ≤ 100

lockCmd

lockCmd

c ≤ 100
Previous work
Formal Requirements Engineering with Modal Sequence Diagrams (MSDs)

- Modal Sequence Diagrams (MSDs)
  - UML-compliant variant of Live Sequence Charts
  - Formal and modal semantics

- Analysis techniques
  - Formal verification on consistency
  - Simulative validation by means of Play-out algorithm

Provisional message
Mandatory messages
Timing requirement

© Fraunhofer IPT / Heinz Nixdorf Institute
Sheet 8
Previous work
Simulative Validation by Means of Play-out Algorithm

- Operational interpretation of formal semantics of MSDs
- „Plays out“ messages specified by the scenarios
- Detects safety violations
- Allows to validate for unintended behavior
Previous work
Simulative Validation by Means of Play-out Algorithm

- Operational interpretation of formal semantics of MSDs
- „Plays out“ messages specified by the scenarios
- Detects safety violations
- Allows to validate for unintended behavior
Previous work
Simulative Validation by Means of Play-out Algorithm

- Operational interpretation of formal semantics of MSDs
- „Plays out“ messages specified by the scenarios
- Detects safety violations
- Allows to validate for unintended behavior

msd RemoteLockDoors

rk: RemoteKey  bcm: Body ControlModule  dl_driver: DoorLock

lockCenReq  lockCmd

Cut = state of an active MSD

msd LockingFeedback

bcm: Body ControlModule  dl_driver: DoorLock  tsa_front: Turn SignalActuator

lockCmd  ack  lockFeedbackCmd

Activation and synchronisation of scenarios by message exchange
Previous work
Simulative Validation by Means of Play-out Algorithm

- Operational interpretation of formal semantics of MSDs
- „Plays out“ messages specified by the scenarios
- Detects safety violations
- Allows to validate for unintended behavior

Cut = state of an active MSD

Activation and synchronisation of scenarios by message exchange
Previous work
Simulative Validation by Means of Play-out Algorithm

- Operational interpretation of formal semantics of MSDs
- “Plays out” messages specified by the scenarios
- Detects safety violations
- Allows to validate for unintended behavior
Previous work
Simulative Validation by Means of Play-out Algorithm

- Operational interpretation of formal semantics of MSDs
- “Plays out” messages specified by the scenarios
- Detects safety violations
- Allows to validate for unintended behavior
Previous work
Simulative Validation by Means of Play-out Algorithm

- Operational interpretation of formal semantics of MSDs
- „Plays out“ messages specified by the scenarios
- Detects safety violations
- Allows to validate for unintended behavior

Unintended behavior: speed locking activates locking feedback!
Previous work
Simulative Validation by Means of Play-out Algorithm

- Operational interpretation of formal semantics of MSDs
- „Plays out“ messages specified by the scenarios
- Detects safety violations
- Allows to validate for unintended behavior
CONTENTS

1. Introduction + motivation

2. Preliminaries: Previous approach on formal requirements engineering

3. Problem description:
   Approach not sufficient for hierarchical component architectures

4. Adapted modeling approach

5. Simulative validation of requirements for hierarchical component architectures
Problem Description
How to Specify and Simulate MSDs for Hierarchical Component Architectures?

- Component architectures

Ports + directed connectors

- Until now: Plain class models as structural basis for MSDs
  - Scenarios only at one hierarchy level
  - Each entity can communicate with arbitrary others
CONTENTS

1. Introduction + motivation

2. Preliminaries: Previous approach on formal requirements engineering

3. Problem description:
   Approach not sufficient for hierarchical component architectures

4. Adapted modeling approach

5. Simulative validation of requirements for hierarchical component architectures
Modeling Approach
Modeling Approach

Overview

Port interfaces

Component types

Internal structure

Scenarios

Top level

Subsystem level(s)

Type View

Internal Structure View

Interaction View

Port interfaces

Component types

Internal structure

Scenarios

Top level

Subsystem level(s)

Type View

Internal Structure View

Interaction View
**Modeling Approach**

**Top Level: Specify Port Interfaces + Component Types**

<table>
<thead>
<tr>
<th>bdd System [Interfaces]</th>
</tr>
</thead>
<tbody>
<tr>
<td>«interface» RK2BCM</td>
</tr>
<tr>
<td>lockCenReq()</td>
</tr>
<tr>
<td>«interface» BCM2DL</td>
</tr>
<tr>
<td>lockCmd()</td>
</tr>
<tr>
<td>«interface» DL2BCM</td>
</tr>
<tr>
<td>ack()</td>
</tr>
<tr>
<td>«interface» BCM2TSA</td>
</tr>
<tr>
<td>lockFeedbackCmd()</td>
</tr>
</tbody>
</table>

**Interpreted Diagram:***

- **Interfaces specify, which messages can be sent within MSDs**
- **Interface usages in ports define possible communication directions**
- **Blocks specify the types for the component architecture**

- **Type (provided)**
- **Type (required)**

**Example:**

- **System Blocks**
  - `BodyControl Module`
  - `RemoteKey`
  - `TurnSignal Actuator`
  - `DoorLock`
  - `LockCmd` (required)
  - `LockFeedbackCmd` (provided)

**Example Interface Usages:**

- `lockCenReq()` (used in `TurnSignal Actuator`)
- `lockCmd()` (required in `DoorLock`)
- `lockFeedbackCmd()` (used in `BodyControl Module`)

**Example Interface Specifications:**

- `RK2BCM`
- `BCM2DL`
- `DL2BCM`
- `BCM2TSA`
Modeling Approach
Top Level: Specify Use Cases

- Use cases by means of UML collaboration diagrams
- Specify participants for a particular use case to be used in its scenarios
Modeling Approach
Top Level: Specify Scenarios

Lock Doors Remotely and Give Feedback

Messages sent via connectors

Lifelines represent use case participants

Messages compliant to port interfaces

Top Level: Specify Scenarios

Lock Doors Remotely and Give Feedback

Messages sent via connectors

Lifelines represent use case participants

Messages compliant to port interfaces
Modeling Approach
Subsystem Level(s): Decompose Structural Elements

Lock Doors Remotely and Give Feedback

Hierarchically decompose components

Connect inner components to superordinate level with delegation connectors

Add and connect inner components
Modeling Approach
Subsystem Level(s): Specify Interactions

Connect MSDs with UML InteractionUse

Add MSDs for new hierarchy level

Messages crossing hierarchy levels connected via gates
Syntactical Constraints for Complete and Executable Models

Automatic Checks as Support for the Engineer

All links set?

Interface compatibility?

Connector directions respected?

bdd System [Interfaces]

«interface» RK2BCM
lockCenReq()

...
CONTENTS

1. Introduction + motivation

2. Preliminaries: Previous approach on formal requirements engineering

3. Problem description:
   Approach not sufficient for hierarchical component architectures

4. Adapted modeling approach

5. Simulative validation of requirements for hierarchical component architectures
Hierarchical Play-out
Functional Principle

msd RemoteLockDoorsAndFeedback
rk: RemoteKey
bcm: Body ControlModule
dl_driver: DoorLock
tsa_front: Turn SignalActuator

ref
lockCenReq
lockCmd
ack
lockFeedbackCmd

msd ProcessLockingRequest
lcm: Lockg CenMst
fm: Flash Manager

lockCenReq
lockCmd
ack
lockFeedbackReq
lockFeedbackCmd

refersTo
Hierarchical Play-out

Functional Principle

- **msd RemoteLockDoorsAndFeedback**
  - 
  - **rk**: RemoteKey
  - **bcm**: Body ControlModule
  - **dl_driver**: DoorLock
  - **tsa_front**: Turn SignalActuator
  - **lockCenReq**
  - **lockCmd**
  - **ack**
  - **lockFeedbackCmd**
  - **lockFeedbackReq**

- **msd ProcessLockingRequest**
  - **lcm**: Lockg CenMst
  - **fm**: Flash Manager
  - **lockCenReq**
  - **lockCmd**
  - **ack**
  - **lockFeedbackReq**
  - **lockFeedbackCmd**

- **refersTo**
Hierarchical Play-out

Functional Principle

msd RemoteLockDoorsAndFeedback

rk: RemoteKey

bcm: Body ControlModule

dl_driver: DoorLock

tsa_front: Turn SignalActuator

ref

lockCenReq

lockCmd

ack

lockFeedbackCmd

lockFeedbackReq

refersTo

msd ProcessLockingRequest

lcm: Lockg CenMst

fm: Flash Manager

lockCenReq

lockCmd

ack

lockFeedbackReq

lockFeedbackCmd
Hierarchical Play-out

Functional Principle

msd RemoteLockDoorsAndFeedback

- rk: RemoteKey
- bcm: Body ControlModule
- dl_driver: DoorLock
- tsa_front: Turn SignalActuator

ref

- lockCmd
- lockFeedbackCmd
- lockCenReq
- lockFeedbackReq

refersTo

msd ProcessLockingRequest

- lcm: Lockg CenMst
- fm: Flash Manager

- lockCenReq
- lockCmd
- lockFeedbackReq
- lockFeedbackCmd
Hierarchical Play-out
Functional Principle
Hierarchical Play-out

Functional Principle
Hierarchical Play-out
Functional Principle
Hierarchical Play-out

Functional Principle

msd RemoteLockDoorsAndFeedback

bcm: Body ControlModule
dl_driver: DoorLock
tsa_front: Turn SignalActuator

rk: RemoteKey
dl_driver: DoorLock

lockCenReq

lockCmd

ack

lockFeedbackCmd

msd ProcessLockingRequest

lcm: Lockg CenMst
fm: Flash Manager

lockCenReq

lockCmd

ack

lockFeedbackReq

lockFeedbackCmd

ref

refersTo
Starting Situation
Formal Requirements and Hierarchical Component Architectures

Previous work

- Formal, scenario-based requirements engineering approach
- Not suited for hierarchical component architectures!

How to specify formal requirements for hierarchical component architectures across several abstraction levels?
Summary and Outlook

- Outlook
  - Verification of correct refinement
  - Cover AUTOSAR specifics
  - More detailed timing information
  - Alignment with systems engineering

Enables integrated requirements engineering and component architecture design across multiple abstraction levels!
Vielen Dank für Ihre Aufmerksamkeit

Dipl.-Inform. Jörg Holtmann
Fraunhofer-Institut für Produktionstechnik
Projektgruppe Entwurfstechnik Mechatronik
Zukunftsmeile 1
33102 Paderborn
Telefon: +49 5251 5456-271
Fax: +49 5251 5465-102
Joerg.Holtmann@ipt.fraunhofer.de
www.ipt.fraunhofer.de/mechatronik

Dr. Matthias Meyer
Fraunhofer-Institut für Produktionstechnik
Projektgruppe Entwurfstechnik Mechatronik
Zukunftsmeile 1
33102 Paderborn
Telefon: +49 5251 5456-122
Fax: +49 5251 5465-102
Matthias.Meyer@ipt.fraunhofer.de
www.ipt.fraunhofer.de/mechatronik