Fail-Safety Checking for Interactive Services

Ingolf H. Krueger
http://sosa.ucsd.edu

Presentation at ASOSA Workshop, June 16, 2008

based on joint work with Vina Ermagan, Emilia Farcas, Claudiu Farcas and Massimiliano Menarini
Motivation

• Embedded systems permeate society
• High quality demands
• Compounded by increasing importance of distributed “systems of systems”
• Challenge: Scattering of functionality
• Transition from component-centric to end-to-end system views; keyword: services
• Failure management is an end-to-end activity
  • Development process dimension
  • Runtime dimension
Service Engineering Overview

1. Conceptual Framework for Services and Components
2. Hierarchical Architecture Pattern for Service-Oriented Architectures (SOAs)
3. Description Techniques for Services and Components
4. Integrated Development Process and Tool Support
Example: Central Locking System

all doors unlock after an impact
Challenges

• Component-level vs. service-level failures
• Inadequate syntax & semantics for interaction specifications
• Failure models are part of neither interaction nor architecture models
• No decoupling of logical and deployment/implementation architecture
• No end-to-end tool support at the architectural level
Contributions

• Focus on **interactions** as unit of modularity
  → service specifications
• Improved syntax and semantics for interaction specifications
• Explicit, scalable models for failures, unmanaged and managed services
• Novel Architecture Definition Language (ADL) integrating services, logical and deployment architectures with failure models
• Transformation from ADL to model checker for automated fail-safety checking
Fail-Safety Checking: Overview

1. Interaction Model

2. Detectors/Mitigators
   - if unlocking fails
   - then automatically inject the unlock message to LM

3. Deployment Model

4. Failure Hypothesis
   - CAN-Bus loses at most one message
   - At most one concurrent component failure
   - ...

5. Model-Checking
   - Safety property holds even if failures occur according to the failure hypothesis
## Terminology

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault</td>
<td>Adjudged or hypothesized cause of an error</td>
</tr>
<tr>
<td>Error</td>
<td>Incorrect system state reached as a consequence of a fault</td>
</tr>
<tr>
<td>Failure</td>
<td>Undesirable, observable consequence of an error</td>
</tr>
</tbody>
</table>
# Terminology

<table>
<thead>
<tr>
<th>Term</th>
<th>Example: Incorrect Airbag Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault</td>
<td>“Bug”: Variable that stores which airbags are activated is overwritten during airbag deployment</td>
</tr>
<tr>
<td>Error</td>
<td>Previously deactivated airbag is activated</td>
</tr>
<tr>
<td>Failure</td>
<td>Supposedly deactivated airbag gets deployed during crash</td>
</tr>
</tbody>
</table>
Overview

• Motivation and Problem Definition
• System, Component and Service Model
• Interaction Model
• Architecture Definition with Failure Models
• Fail-Safety Checking with Promela/SPIN
• Summary and Outlook
System Model

channel

role/component
System Model

Fail-Safety Checking for Interactive Services
System Model

\[ C \]
Channel set

\[ M \]
Message set

\[ \tilde{X} = \mathcal{O}(\mathbb{N} \rightarrow (X \rightarrow M^*)) \]
Valuation of channels in \( X \)

\[ P \]
Component/role set

\[ S_p \]
States of component \( p \)

\[ S = \prod_{p \in P} S_p \]
System component state

\[ \tilde{S} = \mathcal{O}(\mathbb{N} \rightarrow S) \]
State valuation
Causality

$Q$ is causal

$\equiv$

Outputs at time $t$ depend at most on inputs received strictly before time $t$. 
Component and Service Model

For a syntactic interface \( I \triangleright O \) we call
\[
Q : \bar{I} \rightarrow \emptyset(\bar{O})
\]

**Component**, if the input/output relation is **causal**

**Service**, if the input/output relation is **causal** over
\[
Dom.Q = \{ x \in \bar{I} : Q.x \neq \emptyset \}
\]

Components are **left-total**, services are **partial**
input/output relations
Overview

- Motivation and Problem Definition
- System, Component and Service Model
- Interaction Model
- Architecture Definition with Failure Models
- Fail-Safety Checking with Promela/SPIN
- Summary and Outlook
Basic MSC

service unlocking_1

KF

kc:unlck

c:ok

CTRL

clm:unlck

LM

lmc:unlck_ok

LS

cls:sig_unlck
Basic MSC

service unlocking_2

KF

CTRL

SM

Tuner

kc:unlck → csm:handle_id → smk:get_id

ksm:id

par

ck:ok → smc:id_ok

smt:id
HMSCs: Join

**service unlocking**

**join**

- unlocking_1
- unlocking_2

**service unlocking_1**

- KF
  - kc:unlck
  - ck:ok

- CTRL
  - clm:unlck
  - lmc:unlck_ok
  - cls:sig_unlck

- LM
- LS

**service unlocking_2**

- KF
- CTRL
- SM
- Tuner
- par
- ck:ok
- smc:id_ok
- smt:id

- kcf:unlck
- csm:handle_id
- smk:get_id
- ksm:id
- ksm:id

Fail-Safety Checking for Interactive Services
HMSCs: Sequencing, Alt, Loops

```
service basic_cla
```

- unlocking
- locking
MSC Semantics

\[ \varphi \in (\vec{C} \times \vec{S}) \]

**Service** impact

- **IS**
- **CTRL**
- **LM**

- isc:impact
- clm:unlck
- lmc:unlck_ok

- \( u \)
- \( t \)
Semantics - Flavor

Message:

\[
\begin{aligned}
\left[ ch:m \right]_u & \triangleq \left\{ (\varphi,t) \in (\bar{C} \times \bar{S}) \times \mathbb{N} : \\
t > u & \text{ is minimum time at which } m \text{ occurs on } ch \right\}
\end{aligned}
\]

Alt:

\[
\left[ \alpha;\beta \right]_u \triangleq \left[ \alpha \right]_u \cup \left[ \beta \right]_u
\]

Sequencing:

\[
\begin{aligned}
\left[ \alpha;\beta \right]_u & \triangleq \left\{ (\varphi,t) \in (\bar{C} \times \bar{S}) \times \mathbb{N} : \\
\exists t' & \in \mathbb{N}_\infty \because (\varphi,t') \in \left[ \alpha \right]_u \land (\varphi,u) \in \left[ \beta \right]_{u'} \right\}
\end{aligned}
\]
Other Language Elements

- Guarded MSCs
- Bounded, unbounded repetition
- Preemption, preemptive loop
- Trigger Composition
- References
- Local actions
- Timers
Overview

• Motivation and Problem Definition
• System, Component and Service Model
• Interaction Model
• Architecture Definition with Failure Models
• Fail-Safety Checking with Promela/SPIN
• Summary and Outlook
Fail-Safety Checking for Interactive Services

ADL with Failure Models

**Interaction Model**

```
if unlocking fails
then automatically inject the unlock message to LM
```

**Deployment Model**

**Service ADL**

**Detectors/Mitigators**

**Failure Hypothesis**

- CAN-Bus loses at most one message
- At most one concurrent component failure
- ...
Service-Detector-Mitigator Model

MSC

Service --> Unmanaged Service --> Interaction Specification

* operand

Atom

Composite
Service-Detector-Mitigator Model

[Diagram showing the model with nodes labeled as Service, Managed Service, Fault Detector, Mitigator, Detection Strategy, Mitigation Strategy, Unmanaged Service, Interaction Specification, Atom, Composite, and Detectors/Mitigators.]

Detectors/Mitigators
Unmanaged Service

**service** impact

- **IS**
  - isc:impact

- **CTRL**
  - clm:unlck
  - lmc:unlck_ok

- **LM**
Fail-Safety Checking for Interactive Services

Managed Service

managed service m_impact manages service impact
detector

mitigator
Fail-Safety Checking for Interactive Services

Managed CLS

```
service cls
  join
  basic_cls
  m_impact
```
Failure Hypothesis and Deployment Mapping

**component type:** Impact_Sensor  
**plays:** IS  
**can_fail:** true

**component type:** Lock_Motor  
**plays:** LM  
**can_fail:** false

**connection type:** CANBus-Connection  
**plays:** clm, cls, csm, isc, lmc, mlm, smc, smt  
**can_fail:** false  
**can_lose_msg:** true
Failure Hypothesis and Deployment Mapping

ImpactSensor1: Impact_Sensor
ImpactSensor2: Impact_Sensor

LockMotor: Lock_Motor
Controller: Control_ECU

CAN: CANBus-Connection
Overview

• Motivation and Problem Definition
• System, Component and Service Model
• Interaction Model
• Architecture Definition with Failure Models
• Fail-Safety Checking with Promela/SPIN
• Summary and Outlook
Fail-Safety Checking

Service ADL

Interaction Model

Detectors/Mitigators

Deployment Model

Failure Hypothesis

Model Checker: SPIN

Question: fail-safe?

yes

no + counter-example
Generated State Machines (1/3)
Generated State Machines (2/3)

CTRL

... → ? isc:impact → ! clm:unlock → ? lmc:unlock_ok → ...

Adding sink state simulates component failure

... → ? isc:impact → ! clm:unlock → ? lmc:unlock_ok → ...

CTRL
Generated State Machines (3/3)

Weaving-in mitigation strategy

LM

... -> ? clm:unlck

... -> ! lmc:unlck_ok

... -> ! lmc:unlck_ok

LM

... -> ? clm:unlck

... -> ?.mlm:unlck

... -> ! lmc:unlck_ok
Tool Prototype

Service ADL

FSADL2Promela

• Parser
• Static checking
• State machine synthesis
• Detector/Mitigator weaving
• Promela code generator
• AST Visualizer

SPIN
Overview

• Motivation and Problem Definition
• System, Component and Service Model
• Interaction Model
• Architecture Definition with Failure Models
• Fail-Safety Checking with Promela/SPIN
• Summary and Outlook
Summary

- Complexity of distributed integration architectures demands systematic development approach.

- Fail-Safety is a key property of complex, software-intensive systems.

- Addressing fail-safety requires an end-to-end approach:
  - Development-Time
  - Runtime

- Presented today:
  - Precise interaction model and notations
  - Novel ADL incorporating detector/mitigator specs
  - Fail-safety checking using Promela/SPIN
Outlook

• Extensions
  • Dataflow
  • Detection/Mitigation strategies
  • Refined weaving techniques

• Runtime verification [RV07]
  • Aspect-oriented approach to runtime verification

• Runtime failure management [SCC08]
  • Embedding into Rich Services pattern
  • Fault tolerance approach for enterprise applications

• Extensions to other cross-cutting concerns, such as
  • QoS
  • Authorization/Authentication
  • Policies
Literature

[MBEES08]

[MODELS07]

[RV07]

[SCC08]

[TOSEM07]

Further publications: http://sosa.ucsd.edu/