Challenges in the Modeling and Quantitative Analysis of Safety-Critical Automotive Systems

University of Konstanz

Florian Leitner-Fischer
florian.leitner@uni-konstanz.de

26.03.2011 @ ROCKS 2011
Joint work with

Matthias Kuntz and Bernd Reh
TRW Automotive

Stefan Leue
University of Konstanz
Safety-Critical Automotive Systems

Safety Analysis Methods in the Automotive Domain

Probabilistic FMEA

Probabilistic Analysis of System Architectures

Conclusion
Results of HARA:
- Safety Goals (top level safety requirements)
- Automotive Safety Integrity Level (ASIL)

Demonstrate (ASIL-adjusted) compliance with safety requirements by adequate:
- procedures
- architecture
- design, …

ISO 26262 defines appropriate means for achieving sufficient functional safety
Automotive Safety Integrity Levels (ASIL)

- ASIL is assigned to each function of the system based on hazard analysis and risk assessment
- Defines target values for random hardware failures

<table>
<thead>
<tr>
<th>Level</th>
<th>Random HW Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASIL D</td>
<td>$10^{-8} \text{ h}^{-1}$</td>
</tr>
<tr>
<td>ASIL C</td>
<td>$10^{-7} \text{ h}^{-1}$</td>
</tr>
<tr>
<td>ASIL B</td>
<td>$10^{-7} \text{ h}^{-1}$</td>
</tr>
<tr>
<td>ASIL A</td>
<td>$10^{-6} \text{ h}^{-1}$</td>
</tr>
</tbody>
</table>
Safety Analysis Methods in the Automotive Domain

**Qualitative Methods**
- „identify Failures“
- Qualitative FMEA
- Qualitative Fault Tree Analysis
- Event Tree Analysis

**Quantitative Methods**
- „predict frequency of failures“
- Quantitative FMEA
- Quantitative Fault Tree Analysis
- Event Tree Analysis
- Markov models
- Reliability block diagrams
Safety Analysis Methods in the Automotive Domain

Qualitative Methods

- "identify Failures"
- Qualitative FMEA
- Qualitative Fault Tree Analysis
- Event Tree Analysis

Quantitative Methods

- "predict frequency of failures"
- Quantitative FMEA
- Quantitative Fault Tree Analysis
- Event Tree Analysis
- Markov Models
- Reliability Block Diagrams
Failure Modes and Effects Analysis (FMEA)

FMEA Roadmap (Design & Process)

Step 1
Describe Product or Process

Step 2
Define Functions

Step 3
Identity Potential Failure Modes

Step 4
Describe Effects of Failure

Step 5
Determine Causes

Step 6
Detection Methods/Current Controls

Step 7
Calculate Risk

Step 8
Take Action

Step 9
Assess Results
Disadvantage of FMEA

- Manual Process

- Considers only single sources of failures, i.e.,
  - does not take into account that a system can fail only in presence of several (independent) failures

- Does not take advantage of system architecture / behavioural system description:
  - fault injection!
Probabilistic FMEA

Probabilistic FMEA

♦ Automated approach
  ‣ Multiple Faults can be taken into account
  ‣ Check which failure modes can lead to violation of safety goal
  ‣ System architecture in PRISM
  ‣ Counterexamples as sets of paths:
    - graphical representation possible

♦ Disadvantages:
  ‣ PRISM model is input of analysis:
    - SysML / UML are industrial standard, used for real-world application
State of the Art: Analysis of SysML / UML Models

SysML / UML Model

Failure Mode Specification

(Safety-) Requirements

PRISM Model

CSL Property

\[ P_{=?}[(\text{true})U\leq T(\text{inadvertent-deployment})] \]

Manual

Error prone
+ Time consuming

Prob. counterexample

Large number of paths in counterexample,
Interpretation on level of PRISM model
The QuantUM Approach

- Failure Mode Specification
- (Safety-) Requirements

QuantUM UML Model

UML Sequence Diagrams

Fault Trees

PRISM Model

CSL Property
\[ P_{\text{true}}[\text{adventent\_deployment}] \leq T \]

Prob. Counterexample

Automatic

The QuantUM Tool

UML CASE Tool (e.g. IBM Rational Software Architect)

QuantUM

- PRISM
- DiPro
- Fault Trees

UML Sequence Diagrams
The QuantUM Approach

- Failure Mode Specification
- (Safety-) Requirements
- QuantUM UML Model
  - automatic
- UML Sequence Diagrams
  - automatic
- Fault Trees
  - automatic
- PRISM Model
  - automatic
- CSL Property
  \[ P_{\omega}[(true)U_{<=T}(\text{ inadvertent deployment})] \]
- Prob. Counterexample
  - automatic
The QuantUM Model

♦ Requirements for the QuantUM Extensions:

- Based on
  - „Basic concepts and taxonomy of dependable and secure computing“ Avizzienis et al. (2004)

- Derived requirements
  - Applicable to system and software architectures
  - Specification of
    - Safety requirements
    - dependability characteristics (failure modes, …)
    - failure propagation and component dependencies
    - safety mechanisms (repair management, redundancy structures)
  - Experienced user shall be able to use QuantUM with minimal training
  - Additional modeling cost shall be kept minimal
The QuantUM Model

♦ QuantUM Profile

♦ Extension of the UML with Stereotypes

♦ Allows for the specification of
  - Safety Requirements
  - Dependability characteristics (failure modes, …)
  - Failure propagation and component dependencies
  - Safety mechanisms (repair management, redundancy structures)

♦ Used within the UML CASE Tool
  - hence low training costs

♦ Annotation directly in the architecture model
  - hence low additional modeling costs
The QuantUM Model

♦ QuantUM Profile: QUMComponent

- Stereotype that can be used to tag
  - Classes
  - Components

- Associates UML elements with
  - A state machine representing normal behavior
  - (Multiple) state machines representing failure modes
  - A list of (failure) rates that can be used in the state machines
The QuantUM Model

♦ QuantUM Profile: Normal Behavior and Failure Modes

- State machines are used to describe the
  - Normal behavior of QUMComponents
  - Failure Modes of QUMComponents

- Each QUMComponent is associated with
  - 0..1 normal behavior state machines
  - 0..n failure pattern state machines describing failure modes

- Transitions are extended by stereotypes to specify
  - Exponential distributed rates
The QuantUM Model

♦ QuantUM Profile: QUMStateConfiguration

- Goal
  - What is the probability of reaching certain states of the UML model?
  - States need to be somehow identified

- Solution: QUMStateConfiguration
  - States of the UML state machines can be associated with a name and the “and” or “or” operator
  - All states tagged with the same name are forming a state formula connected by the define operator
    - E.g. state1 and state2, state2 or state3
  - The state formula can then be used in the CSL formula
Translation to PRISM

- Failure Mode Specification
- (Safety-) Requirements
- QuantUM UML Model
- PRISM Model
- CSL Property
  \[ P_{\text{true}} (\text{madvertent\_deployment}) \]
- UML Sequence Diagrams
- Fault Trees
- Prob. Counterexample
Fault Tree Generation

- Failure Mode Specification
- (Safety-) Requirements
- QuantUM UML Model
- UML Sequence Diagrams
- PRISM Model
- CSL Property
  \[ P = \Pr[(\text{true}) \land (\text{madvertent\_deployment})] \]
- Fault Trees
- Automatic Prob. Counterexample
Fault Trees

- Used to illustrate error causality
- Engineering practice method (IEC61508, ISO26262, DO-178B)

- Fault Tree Elements

- Basic Event
- OR-Gate (Disjunction)
- AND-Gate (Conjunction)
- Priority AND-Gate (Conjunction with Order)
- Intermediate Event
Fault Tree Generation

♦ Mapping of Counterexamples to Fault Trees

- Computation of the fault tree
  - Computation of combinations of basic events causing the hazard
  - Represents events that are causal for the hazard
  - Computational complexity
    - NP-complete
    - Therefore: over approximation preserving all causality

- Result: compact representation preserving causal information
  - Hundreds of paths in the counterexample
  - Few gates in the fault tree

- More details: technical report “soft-11-02”
  www.se.inf.uni-konstanz.de > Publications > „From Probabilistic Counterexamples via Causality to Fault Trees“
Example: Airbag System - "Inadvertent Deployment"
Mapping Counterexamples to UML

Failure Mode Specification

(Safety-) Requirements

QuantUM UML Model

UML Sequence Diagrams

Fault Trees

Prob. Counterexamples

PRISM Model

CSL Property

\[ P_{=?}\left[ (true) U_{=?} T( inadvertent\_deployment) \right] \]
Example: Airbag System - „Inadvertent Deployment“

![Diagram of Airbag System with UML elements and transitions]

- **Inadvertent deployment**: 0.002336
- **Microcontroller Failure**: 2.44771128118857E-4
- **Enable FET**: 0.00208620
- **Arm FAS**: 0.00208620
- **Fire FAS**: 0.00208620

Transitions:
- 1: Transition (Normal Operation, Microcontroller Failure)
- 1.1: Enable FET
- 1.1.1: Transition (Disabled, Enabled)
- 1.2: Enable FET
- 1.3: Arm FAS
- 1.3.1: Transition (Idle, Armed)
- 1.4: Arm FAS
- 1.5: Fire FAS
- 1.5.1: Transition (Armed, Fired)
- 1.6: Fire FAS
Evaluation

♦ Airbag Control System

♦ and other case studies outside of the automotive domain.
Evaluation of the QuantUM Approach

♦ Results:

♦ Model translation from UML to PRISM
  - Before: several man-days of highly trained engineers
  - Now: automatic translation in a few seconds

♦ Mapping the Counterexample to fault trees and UML sequence diagrams aids interpretation
  - Counterexamples with 500-750 paths can be reduced to fault trees with 5-10 gates
  - Patterns can be found immediately
    • Before comparison of several hundred paths was required
  - UML Sequence diagrams
    • Good representation for failure propagation by method calls
Conclusion

♦ QuantUM
  ♦ Automatic safety analysis of UML / SysML models
    - Automatic translation to PRISM
    - Transformation of probabilistic counterexamples to fault trees
    - Representation of probabilistic counterexamples as UML sequence diagrams

♦ Future Work
  ♦ Extension to other architecture description languages

  ♦ Tighter integration in industrial tool chains
    - Requirements Management
    - Configuration and Version Management
Questions ...