Quantitative Safety Analysis of Non-Deterministic System Architectures

Adrian Beer

University of Konstanz
Department of Computer and Information Science
Chair for Software Engineering

Adrian.Beer@uni.kn
Motivation

- Safety critical systems are everywhere

- These systems have to be verified against safety goals to ensure safe working
  - Safety analysis should be easily supported during the development!
  - Best case: completely automatized
Outline

1. Motivation

2. Preliminaries

3. Safety Analysis of UML / SysML models
   - The QuantUM approach

4. Case Studies

5. Conclusion
Quantitative Safety Analysis of Non-Deterministic System Architectures
Quantitative Safety Analysis of Non-Deterministic System Architectures

- Industrial Practice (some demanded by safety standards)

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

- Academia

Model Checking
Probabilistic Model Checking
How is non-determinism introduced in systems?

- Environmental behavior
  - No probability for environmental factors
  - Can happen non-deterministically at any point in time

- Concurrency
  - Several processes / components run concurrently
  - Scheduler resolves non-determinism by deciding which process is allowed to take the next step

- Abstraction
  - Some unknown aspects during design / modeling phase
  - “Incompleteness” of the design model
  - Simplification / abstraction of certain aspects in the system
Quantitative Safety Analysis of Non-Deterministic System Architectures

♦ Model-based Engineering
  ‣ Models help to structure, develop, analyze complex systems

♦ Model-based Engineering promoted / demanded by modern standards
  ‣ ISO 26262
  ‣ DO-178C
  ‣ ARP 4754A
  ‣ ESAAR4

♦ Modeling languages
  ‣ UML / SysML
  ‣ Matlab Simulink
  ‣ AADL
  ‣ ASCET
  ‣ …
Outline

1. Motivation
2. Preliminaries
3. Safety Analysis of UML / SysML models
   - The QuantUM approach
4. Case Studies
5. Conclusion
The QuantUM Approach

The Goal:

- Automatic verification of UML / SysML models easily applicable and consistent in industrial practice
- Safety related information is directly encoded in the model using stereotypes
  - Normal + failure behavior
  - Quantitative information, i.e. failure rates
  - Safety requirements encoded in state configurations of the system
    - Automatic translation into reachability properties
The Goal:

- Automatic verification of UML / SysML models easily applicable and consistent in industrial practice
The QuantUM Approach

- **QuantUM** relies on the concept of **model checking**
  - Automatic exploration of the state space of the model of a system
    - **PRISM model checker**
      - Probabilistic analysis
    - **SPIN model checker**
      - Functional analysis
  - Systematic search for modeling flaws in the system
The Problem:

- Model of computation until now: Continuous Time Markov Chains
  - Only stochastic transitions
  - Modeling trick:
    - Non-determinism is approximated using pseudo-stochastic transitions
    - Introduced error often very large
The QuantUM Approach

Example:

CTMC: "pseudo-stochastic" transition failure transition

\[ \lambda = \frac{1}{h} \quad 10^{-6}/h \]

\[ S_0 \xrightarrow{\lambda = 1/h} S_1 \xrightarrow{10^{-6}/h} S_2 \]

- Probability of reaching state \( S_1 \) within 1h is \( \approx 0.63 \)
  - Expectation: reaching state \( S_1 \) within 1h should always give a probability of 1

- Even when setting \( \lambda \) to a higher value this phenomenon has an impact along long paths
The QuantUM Approach

Solution: Use Markov Decision Processes

- MDPs support non-determinism by definition
- MDPs have a discrete time-basis
  - No continuous failure rates are supported by MDPs
  - Discretization is possible: Approximation of continuous negative exponential distribution with a discrete geometric distribution

- Introduced error is computable and orders of magnitude smaller than the actual value

- Discretization step size has a significant effect on computation time
How is the translation done?

The QuantUM Approach

UML or SysML Model → UML Semantics → MDP → PRISM Code

→ same semantic foundation
Outline

1. Motivation
2. Preliminaries
3. Safety Analysis of UML / SysML models
   ▪ The QuantUM approach
4. Case Studies
5. Conclusion
Case Studies

- Airbag System
- Airport Surveillance Radar
Example: Airbag System

◆ **UML Model of an Airbag System**

- **MainSensor**
  - acceleration : Integer
  - mainSensor

- **SafetySensor**
  - acceleration : Integer
  - safetySensor

- **MicroController**
  - criticalCrashLevel : Integer
  - criticalCrash : Boolean
  - transition(“NormalOperation”, “MicroControllerFailure”)()

- **FET**
  - fetEnabled : Boolean
  - enableFET()
  - transition(“Disabled”, “Enabled”)()
  - transition(“FETNormalBehavior”, “FETStuckHigh”)()

- **FASIC**
  - fasicArmed : Boolean
  - fasicFired : Boolean
  - armFASIC()
  - fireFASIC()
  - transition(“FasicNormalBehavior”, “FASICShortage”)()
  - transition(“Idle”, “Armed”)()
  - transition(“Armed”, “Fired”)()
  - transition(“FASICNormalBehavior”, “FASICStuckHigh”)()

◆ **Computation of** „Probability of an inadvertent deployment within 100h”
Example: Airbag System

- Statechart of the Microcontroller

![Statechart Diagram](image-url)
Example: Airbag System

PRISM Code

module MicroController

NormalOperation_active: [0..19] init 0;

// initial state
[] (NormalOperation_active = 0) -> NormalOperation_active '!= 1);

[] (NormalOperation_active = 6) & (MicroController_criticalCrashLevel >= 3)
  -> (NormalOperation_active '!= 7) & (MicroController_criticalCrash 'true);

dendmodule
Example: Airbag System

- **C Code**

```c
switch (NormalOperation_active) {
    ........ // some code
    case EvaluationDone:
    {
        if(IS_EVENT_TYPE_OF(OMNullEventId))
        {
            //## transition 2
            if(criticalCrash = false)
            {
                EvaluateCrash_exit();
                NormalOperation_subState = Idle;
                rootState_active = Idle;
                res = eventConsumed;
            }
        }
        if(res == eventNotConsumed)
        {
            res = EvaluateCrash_handleEvent();
        }
    }
    break;
    ........ // some code
}
```
Evaluation

- Computation of failure probabilities for the inadvertent deployment

<table>
<thead>
<tr>
<th></th>
<th>CTMC $\lambda = 1$</th>
<th>CTMC $\lambda = 100$</th>
<th>MDP (non-det.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airbag (probability)</td>
<td>$2.0 \cdot 10^{-4}$</td>
<td>$2.7 \cdot 10^{-4}$</td>
<td>$9.98 \cdot 10^{-4}(\pm 8.32 \cdot 10^{-11})$</td>
</tr>
<tr>
<td>Airbag (time)</td>
<td>0.1 sec.</td>
<td>258.1 sec.</td>
<td>3.94 sec.</td>
</tr>
<tr>
<td>Radar (probability)</td>
<td>$8.8 \cdot 10^{-22}$</td>
<td>$8.231 \cdot 10^{-20}$</td>
<td>$4.81 \cdot 10^{-13}(\pm 1.39 \cdot 10^{-20})$</td>
</tr>
<tr>
<td>Radar (time)</td>
<td>22.57 min</td>
<td>68.88 min</td>
<td>277.27 min</td>
</tr>
</tbody>
</table>

- ASR: “Probability of wrong information being displayed to the air traffic manager within 1h”
- Model sizes:
  - Airbag: $\approx 7000$ states + 50,000 transitions
  - ASR: $\approx 200$ mio. states + 2 billion transitions
Conclusion

◆ Summary: QuantUM Approach
  ▶ Quantitative model-based safety analysis
  ▶ Automatic translation of UML / SysML models into model checking code
  ▶ Non-determinism + continuous failure rates can now be handled while maintaining the computation error
  ▶ Computation is adaptable to the purposes of the results
    – Certification or just coarse evaluation of design

◆ Outlook
  ▶ Automatic Fault Tree generation for MDPs
  ▶ Automatic Failure Mode and Effect Analysis
  ▶ Result interpretation as UML sequence diagrams
  ▶ Further integration into certification and validation standards
    – ISO26262, ARP 4754A