Testing Applications of Cyber-Physical Systems in the Presence of Uncertainty

Martin A. Schneider
Fraunhofer FOKUS, Berlin, Germany
October 10th, STV Workshop, Berlin, Germany
Project facts:

Total cost: EUR 3 713 233,75
EU contribution: EUR 3 713 233,75
Coordinator: Oslo Medtech, Norway
Topic(s): ICT-01-2014 - Smart Cyber-Physical Systems
Funding scheme: RIA - Research and Innovation action

Overall project objective:
Improving CPS dependability via systematic and automated testing of Uncertainty in CPS

The consortium
Results and methods

**Key expected results:**
- Understanding Uncertainty (U-Taxonomy)
- Modeling Framework
  - Extensible and Configurable
- Testing Framework
  - Extensible and Configurable
- Tools implementing Taxonomy and Frameworks
- Standards (Crosscutting)

**Model-Based Testing:**
- Abstraction
- Managing Complexity
- Automation
- Systematic

**Search-Based Testing**
- Optimization
- Smart Mechanisms
- Discovering unknown uncertainties
- Genetic Algorithms.....
Socrates

»I know that I know nothing«

»I know that I don’t know«

»I know that I don’t know with certainty«

© Photograph by Greg O'Beirne. Cropped by User:Tomisti / Wikimedia Commons / CC-BY-SA-3.0 / GFDL
Agenda

1. Uncertainty and Cyber-Physical Systems
2. Uncertainty Taxonomy
3. Uncertainty Modelling
4. Uncertainty Testing
Uncertainty

“any deviation from the unachievable ideal of completely deterministic knowledge of the relevant system”

Walker et al. (2003): Defining uncertainty: a conceptual basis for uncertainty management in model-based decision support

„system state of incomplete or inconsistent knowledge such that it is not possible [...] which of two or more alternative environmental or system configurations hold at a specific point”

Uncertainty in Cyber-Physical Systems

• Cyber-physical systems are connected embedded systems that integrate computation, networking and physical processes.

• Uncertainty arises from interaction between
  • elements of the CPS’s infrastructure  Infrastructure Level
  • application(s) and the infrastructure of the CPS  Integration Level
  • humans and the environment with the CPS  Application Level
CPS may be not dependable

- **undesired behaviour** of a CPS is observed at runtime
  - due to uncertainty in the digital x physical environment

**Challenge**
- How to find such scenarios **efficiently** in the infinite and complex space of the scenarios?

**Solution**
- Search algorithms
Use Cases for Uncertainty Testing

Automated Warehouse
- automatically stores and unloads goods
- manual intervention sometimes required
  - handling goods
  - updating database

GeoSports
- automatically tracks all kinds of movements during a match (positioning via triangulation)
- improving performance of athletes
- athlete wears a device that constantly communicates with locating infrastructure

© Fraunhofer FOKUS
Uncertainty and Knowledge

- **Certainty**
  - known known
  - knowledge we are aware of

- **Uncertainty**
  - known unknown
  - things we know that we don’t know them
  - unknown known
  - things we don’t know and are not aware of
  - unknown unknown
  - things we don’t know that knowledge exists

- **Awareness**
  - knowledge exists
Uncertainty and Knowledge

- **Certainty**
  - Known
  - Known

- **Uncertainty**
  - Known
  - Unknown

- **Unknown**
  - Unknown

- **Knowledge**
  - We are aware of
  - Provided by use cases, observed in the field

- **Goal of U-Test**
  - To find such uncertainties

- Awareness
  - Knowledge exists
Uncertainty and Risk

• uncertainty w.r.t. to the occurrence (likelihood) of a risk
• uncertainties do not have a probability assigned
• uncertainty covers positive and negative outcomes while risk focusses on negative outcomes, e.g., threats

• uncertainty as a source of risk
  • uncertain behavior: manifestation of an uncertainty as an behavior of the CPS with a negative impact on its dependability
Agenda

1. Uncertainty and Cyber-Physical Systems
2. **Uncertainty Taxonomy**
3. Uncertainty Modelling
4. Uncertainty Testing
Uncertainty Taxonomy (Excerpt)

- **nature**
  - epistemic
  - aleatoric
- **environment**
  - cyber environment
  - physical environment
- **cause**
  - human behavior
  - natural process
  - technological process
- **impact**
  - direct
  - indirect
  - impacted element

© Fraunhofer FOKUS
Agenda

1. Uncertainty and Cyber-Physical Systems
2. Uncertainty Taxonomy
3. **Uncertainty Modelling**
4. Uncertainty Testing
Uncertainty Modelling

Uncertainty Modelling Framework (UMF)

• State Machines
  • describe the expected input/output behavior of the SUT
  • from the perspective of SUT

  [guard] trigger / effect

• Uncertainties
  • characterization of uncertainties in terms of the UMF
  • that are related to the model
Agenda

1. Uncertainty and Cyber-Physical Systems
2. Uncertainty Taxonomy
3. Uncertainty Modelling
4. Uncertainty Testing
Search-based Uncertainty Testing

- cover known uncertainties described by use case providers
  - by using use case descriptions (state machines)
  - by using information from modelled uncertainties

- discover unknown uncertainties
  - by exploiting information from known uncertainties (coupling effect)
  - by recombining uncertainties

© Fraunhofer FOKUS
Search-based Uncertainty Testing

• genetic algorithm

• **individual**: state machines representing use cases

• **mutation**: applying mutation operators to state machines
  • first generation: apply mutation operators solely based on uncertainty information
  • further generations: increase amount of mutations not related to modelled uncertainties

• **crossover**: combination of uncertainties

© Fraunhofer FOKUS
# Mutation Operators

<table>
<thead>
<tr>
<th>Mutation Operator</th>
<th>Description</th>
<th>Constraints/Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add Transition</strong></td>
<td>Adds a new transition by duplicating an existing one and setting a new source and target state.</td>
<td>Transitions having an initial state as source or a final node as target must not be removed.</td>
</tr>
<tr>
<td><strong>Remove Transition</strong></td>
<td>Completely removes the transition.</td>
<td>Equivalent to ‘Change Guard: replace expression with false’.</td>
</tr>
<tr>
<td><strong>Remove Transition (with State Merge)</strong></td>
<td>Completely removes the transition. Merges the source and target state if the removed transition is the only one connecting them (optional: with the same direction). This avoid mutilated state machines which inhibit generating test cases.</td>
<td>Equivalent to ‘Change Guard: replace expression with false’.</td>
</tr>
<tr>
<td><strong>Reverse Transition</strong></td>
<td>Swaps source and target of the transition.</td>
<td>Transitions having an initial state as source or a final node as target must not be reversed. Optional: Transitions being the only one that connect source and target state must not be removed (optional: with the same direction). This avoid mutilated state machines which inhibit generating test cases.</td>
</tr>
<tr>
<td><strong>Change Source/Target</strong></td>
<td>Move the source or the target of the transition to any other state.</td>
<td>In case the target state of the transition is changed, the target must not be the initial state. In case the source state of the transition is changed, the source must not be the final node.</td>
</tr>
<tr>
<td><strong>Remove Trigger</strong></td>
<td>Transforms the transition to a completion transition.</td>
<td>Equivalent to ‘Change Guard: replace expression with true’.</td>
</tr>
<tr>
<td><strong>Remove Guard</strong></td>
<td>Removes the guard of a transition completely.</td>
<td>Guards and effects are written in C#.</td>
</tr>
<tr>
<td><strong>Remove Effect</strong></td>
<td>Removes the effect of a transition completely.</td>
<td></td>
</tr>
<tr>
<td><strong>Change Trigger Operation</strong></td>
<td>Changes the operation to another one of the same interface of the original operation.</td>
<td></td>
</tr>
<tr>
<td><strong>Change Trigger/Change Effect</strong></td>
<td>- replace expression with true/false</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- negate expression</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- replace subexpression with true/false</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- negate subexpression</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- change logical operator</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- change relational operator</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- change arithmetic operator</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- change set operator</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- change quantifier</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- replace operand</td>
<td></td>
</tr>
</tbody>
</table>

---

*Guards and effects are written in C#.*
Search-based Testing with a Genetic Algorithm

Exploiting modelled uncertainties

Starting point

Mutation

Candidate solutions = state machines

Quality evaluation (fitness evaluation) & selection

Crossover

Recombine uncertainties

Test case generation & execution

Test case generation & execution
Example

Mutation Operator

Remove Trigger

configureTag
configureTag [the tag id must not already exist in the configured tags list] /
add the passed tag id to the list of configured tags

initiated

calibrating

subsequentMounting
mount [not all required are mounted] /
instantiate new locator and increase number of

waiting_for_signal
setPosition /
store the position data in trace file

recording

receivingPositionData
getAllPositions /
return all currently recorded position data

Initial

initializing /
instanitate number of required

|© Fraunhofer FOKUS|
Example

- **configureTag()**
- **mount(locator0, 0)**
- **mount(locator1, 90)**
- **mount(locator2, 180)**
- **mount(locator3, 270)**
- **calibrating()**
- **mount(locator1, 180)**
- **setPosition(1)**
- **getAllPositions()**

**Mutation Operator**

**Add Transition**

© Fraunhofer FOKUS
Search-based Uncertainty Testing: Fitness Factors

• generic, simple, model-based profile for fitness factors
Coverage Criteria

• Traditional Transition Coverage (state machine)

\[
\frac{\text{#transitions}_{\text{covered}}}{\text{#transitions}_{\text{all}}}
\]

• Uncertainty Coverage (model)

\[
\frac{\text{#uncertainties}_{\text{covered}}}{\text{#uncertainties}_{\text{modelled}}}
\]

• Mutation Transition Coverage (state machine)

\[
\frac{\text{#mutations}_{\text{covered}}}{\text{#mutations}_{\text{all}}}
\]

• Known Uncertainty Space Coverage (all generations related to a single uncertainty)

\[
\frac{\text{#mutations}}{\text{#states} \times (\text{#states} - 1) \times \text{#operations} \times 2}
\]
A few, early numbers...

<table>
<thead>
<tr>
<th>Mutation Operator</th>
<th>#Test Cases</th>
<th>#Removed Test Cases</th>
<th>#Remaining Test Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Complete Path</td>
<td>Mutated Transition Coverage</td>
</tr>
<tr>
<td>ChangeTransitionTarget</td>
<td>51</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ChangeTransitionSource</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>RemoveTransition</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AddTransition</td>
<td>51</td>
<td>0</td>
<td>51</td>
</tr>
<tr>
<td>RemoveEffect</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>RemoveGuard</td>
<td>252</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>RemoveTrigger</td>
<td>51</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

[1] Test cases generated by MS SpecExplorer based on the mutated state machines by traversing the state machines.
[2] Test cases generated by MS SpecExplorer do not necessarily end in a final state. Hence, first all complete paths starting from an initial state and ending in a final state are selected in the first stage.
Conclusions & Future Work

- small effort for testers
  - start from functional models (state machines)
  - add declarative uncertainty descriptions
- reduction of search space
  - search is guided by modelled uncertainties
- configurable and extendable
  - by modelled uncertainties
  - and model-based fitness factors
- empirical evaluation on the case studies
Thank you for your attention!

www.u-test.eu

@utesth2020

U-test.eu