SEMINAR »VERIFICATION AND TESTING OF COMPLEX SYSTEMS«

Dr. Günter Kniesel – Uni Bonn
Dr. Michael Gerz – Fraunhofer FKIE

Kickoff Meeting
October 17th, 2016
Günter Kniesel

Institut für Informatik III
Römerstraße 164
53117 Bonn
0228-73-4511
gk@cs.uni-bonn.de

- Software Analysis and Transformation
- Software Design and Architecture
- Model-Driven Software Development
- Domain Specific Languages
- Agile Software Engineering Methodologies
Michael Gerz

Fraunhofer FKIE
Fraunhoferstraße 20
53343 Wachtberg (south of Bonn)
0228-9435-414
michael.gerz@fkie.fraunhofer.de

- **1990 - 1996:** Study of Computer Science, University of Koblenz-Landau, Enhanced courses in computational linguistics
- **10/1996 - 09/2001:** Research Assistant at the Institute for Telematics, Medical University of Lübeck
- **10/2001 - 01/2003:** Research Assistant at the Institute for Telematics e.V., Trier
- **01/2003 - 10/2003:** Research Assistant at the Chair of Prof. Meinel, University of Trier
- **04/2003:** Dissertation at the University of Göttingen
- **since 01/2004:** Head of Group „Interoperability & Testing“ Department on Information Technology for Command and Control, Fraunhofer FKIE
Applied research on all aspects of Defense and Security

- Reconnaissance, communication networks, command & control, cyber defense

»We are working on making our world safer. Our goal is to detect, minimize, and manage life-threatening risks.«

Visit http://www.fkie.fraunhofer.de
You (?)

- Students of Master Programme in Computer Science
- ... with (some) focus on Inform. & Communication Management track
- Max. 8 participants

Background knowledge:
- Software construction, programming
- Logic, mathematical formalisms

Aim:
- Seminar with grade in fulfillment of module requirements
Scope of the seminar

- Techniques for analyzing the correctness of complex (software) systems
- Theoretical foundations for such techniques
- Application of practical tools
Aims of Seminar, Capabilities

- Study scientific literature
  - Information gathering, search and comprehension
  - Retrieve further literature for additional/background information
  - Stay in touch with supervisor
- Write a scientific article (in English)
  - Summarize the content of the provided papers
  - Provide own explanations and examples
  - Write seminar paper (12–15 pages)
- Give a scientific talk (in English)
  - Present findings on the assigned topic, based on paper
  - Give seminar talk (about 40 minutes), answer questions
- Discuss scientific subjects
  - *Active* participation in moderated group discussions
Seminar Organization

- **Assignment of topics**: this week

- **Self-study of literature**: as soon as possible (next two to three weeks)

- **Talks to be held new year** (probably two per meeting)

- **Preparation phase** (dates to be adjusted for holidays):
  - 6 – 8 weeks before talk: First meeting with supervisor (might be at FKIE)
  - at least 3 weeks b.t.: Submission of draft paper
  - at least 10 days b.t.: Distribution of final paper to all participants
  - at least 1 week b.t.: Discussion of presentation with one of us
  - At any time: Discussions on demand

© Fraunhofer FKIE
A Few (Not Completely Random) Remarks

- Agree with supervisor on what should go into essay and talk
- Don’t be shy asking questions (during preparation & discussions)
- Take notes during preparation meetings
- Discuss the structure of paper and presentation in advance
- Use your own words, examples, explanations
- Provide a motivation, don’t just array fact after fact.
- Use notation and terminology consistently (in particular when using different sources)
- Use a spell-checker (please!)
- Adhere to deadlines! When submitting a final version, make sure it is really final
- The main addressees of your presentations are your fellow students.
Introduction To Basic Concepts
Definitions

- Verification
  - “The process of evaluating software to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.” [IEEE Standard 610]
  - “Are we building the product right?”

- Validation
  - “The process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements.” [IEEE Standard 610]
  - “Are we building the right product?”

- Testing
  - “The process of operating a system or component under specified conditions, observing or recording the results and making an evaluation of some aspects of the system or component.” [IEEE Standard 610]
  - Test can unveil the existence of errors but cannot prove their absence!
Verification – Non-Formal Approach

- Detection of specification violations by means of testing
  - Also known as *manual* verification
  - Most common in practice

- Pros
  - Suitable for complex / huge systems
  - No specific training / education in formal methods required
  - Less expensive than formal approach (time & costs)

- Cons
  - Absence of bugs ("correctness") cannot be proven
  - Less amenable to automation
Verification – Formal Approach

- Methods
  - Model Checking
  - Theorem Proving

- Pros
  - Supports correctness proofs (i.e., that certain properties DO hold)
  - Method(s) of choice for highly reliable systems (Ariane 5 crash)
  - High degree of automation
  - Shows limits of automation (→ scientific insight)

- Cons
  - Requires specific training / education
  - Tricky when applied to complex / huge systems
  - In practice (much) more expensive than testing
Model Checking

- Given
  1. Finite state machine M (→ abstract model of the system under test)
  2. Temporal formula F (→ specification of a desired property)

- Model Checking Verification Problem: Show that M semantically entails F:
  
  \[ M \models F \]  
  ("M is a model of F")

- Algorithmic approach: Explore the set of reachable states of M to ensure that F holds

- Termination requires that the set of reachable states is finite
Theorem Proving (1)

- Expressions of the form
  \[ \{F_{\text{pre}}\} P \{F_{\text{post}}\} \] (Hoare calculus)
  - meaning “if property $F_{\text{pre}}$ holds before program $P$ starts, $F_{\text{post}}$ holds after the execution of $P$”
  - $P$ can refer to an entire program or to an atomic action, depending on the unit to be verified

- Calculus consists of *axioms* and *inference rules* which are used to derive $F_{\text{post}}$ based on $F_{\text{pre}}$ and $P$

- Example inference rule:
  \[
  \frac{\{F_{\text{pre}}\} P \{F_{\text{inter}}\}, \quad \{F_{\text{inter}}\} P' \{F_{\text{post}}\}}{\{F_{\text{pre}}\} P; P' \{F_{\text{post}}\}}
  \]
Most theorem provers used to prove program properties are based on variations of Hoare calculus.

Key difference between theorem proving approach and model checking approach (to software verification):
- Theorem provers do not need to exhaustively visit the program’s state space to verify properties.
- ... can reason about infinite state spaces (e.g., state spaces involving complex datatypes and recursion).

Drawbacks:
- Proof system for a system of practical size can be extremely large.
- Generated proofs can be large and difficult to understand.
- Require a great deal of user expertise and effort.
Classification of Test Methods (1)

Static Test

Manual
  - Review
  - Inspection
  - Structured Walkthrough

Automatic
  - Static Code Analysis
  - Formal Verification
  - Anti-Pattern Detection
Classification of Test Methods (2)

- **Dynamic Test**
  - **Type of System**
    - Hardware Test
    - Software Test
  - **Test Result Authority**
    - Conformance Test
    - Interoperability Test
    - Regression Test
  - **Granularity / Stage**
    - Unit Test
    - Integration Test
    - System Test
  - **System Knowledge**
    - Black Box Test
    - Gray Box Test
    - White Box Test
    - Monitoring
Classification of Test Methods (3)

Dynamic Test

Functional Test
Real-Time Test
Performance Test
Load Test
Stress Test
Portability Test
Penetration Test
User Interface Test
Useability Test
Security Test
Classification of Test Methods (4)

Dynamic Test

Test Intensity / Coverage
- Control Flow
  - Statement
  - Condition
  - Path
- Data Flow
  - Defs-/Uses Criteria
  - ...

Test Data Selection
- Exhaustive Test
- Partition Test
- Boundary Value Test
- Random Test
- Mutation Test
Selected Topics
Topic 1: Testing and Test Control Notation 3 (TTCN-3)

- **Subject**
  - Standardized testing language
  - European Telecommunication Standards Institute (ETSI)
  - Supports automated and distributed testing

- **Objectives**
  - Present the key concepts of the test language
  - Develop sample test suite in TTCN-3
    - Use case: Bank withdrawal, account balance request
  - Analyze tool support (Eclipse Titan)

- **References**
  - http://www.ttcn-3.org
  - https://projects.eclipse.org/projects/tools.titan
**Topic 2: Jnario – Executable Specifications for Java**

- **Subject**
  - Testing, specification, and documentation framework
  - Executable unit, integration, and acceptance specifications
  - Orchestration of specifications
  - Developed at BMW Car IT

- **Objectives**
  - Present the key concepts of Jnario
  - Explain how Domain Specific Languages (DSLs) can be defined with Xbase/Xtext
  - Live demonstration based on self-defined use case

- **References**
  - [http://jnario.org/](http://jnario.org/)
  - [https://github.com/sebastianbenz/Jnario/issues/168](https://github.com/sebastianbenz/Jnario/issues/168)
  - [https://github.com/borisbrodski/Jnario/tree/no_xtend_xtext2.9](https://github.com/borisbrodski/Jnario/tree/no_xtend_xtext2.9)
Subject
- SAT-based formal verification (propositional logic)
- How to use SAT solvers for model checking
- Formulating a verification problem as a SAT problem

Prerequisite:
- Knowledge of math. logic & complexity theory

Objectives
- Understand, present & explain new results

References
Subject
- Continuous integration - integrate code early and often into code mainline
- Automated regression testing

Objectives
- Explain how testing can be coupled with continuous integration

References
Topic 5: Testing Grid and Cloud Infrastructures

Subject
- Testing of grid and cloud infrastructures

Objectives
- Present the specific requirements for testing cloud/grid computing environments
- Present work on standardized test framework

References
Topic 6: Test Generation Based on Finite State Machines

- **Subject**
  - Finite State Machines to model specifications
  - Various methods to derive test cases
    - addressing fault models
    - minimizing the size of test cases

- **Objectives**
  - Describe the underlying assumptions and constraints
  - Present different approaches to generate tests based on FSMs
  - Provide examples

- **References**
Topic 7: Model-Based Testing With Spec Explorer

■ Subject
  ■ Testing environment developed by Microsoft Research
  ■ Reactive Systems = Non-deterministic

■ Objectives
  ■ Understand & present key ideas

■ References
Topic 8: Lazy Systematic Unit Testing

- **Subject**
  - Semi-automatic unit test generation
  - “Tests for full conformance to a lazy specification, which is inferred on-the-fly from the code, by static and dynamic analysis, and from hints supplied by the programmer”

- **Objectives**
  - Understand & present approach
  - Showcase »JWalk«

- **References**
  - Journal and conference publications at [http://staffwww.dcs.shef.ac.uk/people/A.Simons/jwalk/](http://staffwww.dcs.shef.ac.uk/people/A.Simons/jwalk/)

- **Note:**
  - Request academic license from author in time!
Topic 9: Mobile Testing as a Service (MTaaS)

- **Subject**
  - Validation of mobile Apps and SaaS applications on mobile web
  - High complexity due to diversity of mobile devices and computational resources

- **Objectives**
  - Provide a survey of the challenges, technologies, approaches, and infrastructures

- **References**
  - A. Malini; N. Venkatesh; K. Sundarakantham; S. Mercyshalinie: Mobile application testing on smart devices using MTAAS framework in cloud. International Conference on Computer and Communications Technologies (ICCCT), 2014
Subject
- Regression testing as an expensive maintenance activity
- Test selection based on control flow graphs for a (modified) program

Objectives
- Present the concepts and provide examples

References:
Topic 11: Practical Application of Model Checking

- **Subject**
  - Use model checking to find serious errors in file systems
  - Find corner-case errors by exploring the system’s state space
  - File systems as a use case:
    - Errors are most serious
    - Hard to test whether a system recovers after any crash

- **Objectives**
  - Introduce basic concepts of model checking
  - Describe the techniques and tools applied
  - Summarize the findings of the use case

- **References**
Way ahead
Reminder: What we expect from you

- Information gathering
  - Summarize the content of the provided papers
  - Retrieve further literature for additional/background information

- A 12–15 pages summary of the topic (in English)

- A 40 minutes presentation of the findings (in English)
  - Followed by a moderated group discussion

- Active participation in group discussions (and regular attendance)
# Topic Assignment

<table>
<thead>
<tr>
<th>Topic #</th>
<th>Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – Testing and Test Control Notation 3 (TTCN-3)</td>
<td></td>
</tr>
<tr>
<td>2 – Jnario – Executable Specifications for Java</td>
<td></td>
</tr>
<tr>
<td>3 – SAT-Based Formal Verification</td>
<td></td>
</tr>
<tr>
<td>4 – Continuous Integration &amp; Automated Testing</td>
<td></td>
</tr>
<tr>
<td>5 – Testing Grid and Cloud Infrastructures</td>
<td></td>
</tr>
<tr>
<td>6 – Test Generation Based on Finite State Machines</td>
<td></td>
</tr>
<tr>
<td>7 – Model-Based Testing With Spec Explorer</td>
<td></td>
</tr>
<tr>
<td>8 – Lazy Systematic Unit Testing</td>
<td></td>
</tr>
<tr>
<td>9 – Mobile Testing as a Service (MTaaS)</td>
<td></td>
</tr>
<tr>
<td>10 – Regression Test Selection</td>
<td></td>
</tr>
<tr>
<td>11 – Practical Application of Model Checking</td>
<td></td>
</tr>
</tbody>
</table>
Schedule

- Meetings take place
  - on Monday, 12.15 – 13.45,
  - Römerstraße 164, room A121

- Kickoff Meeting (today)
  - Assignment of topics

- Preparation phase
  - 24-Oct, 31-Oct, 07-Nov, 14-Nov, 21-Nov, 28-Nov, 05-Dec, 12-Dec, 19-Dec
  - Meetings after consultation

- Presentation phase
  - 09-Jan, 16-Jan, 23-Jan, 30-Jan, 06-Feb, 13-Feb
  - Possibly two presentations per session

Red = M. Gerz not available