Software and Systems Engineering

Prof. Dr. Stefan Leue

(based on a presentation by D. Boetius und R. Senn)

http://sen.uni-konstanz.de/

Ringvorlesung, WS 2023/2024
Chair for Software and Systems Engineering

♦ Head of Chair:
  ‣ Prof. Dr. Stefan Leue

♦ Secretary:
  ‣ Martina Thibaut

♦ Research Staff
  ‣ David Boetius (Formal Methods for Neural Networks)
  ‣ Raffael Senn (Real-Time Repair)

♦ Student Assistants
  ‣ Jannick Strobel
  ‣ Eric Tepper
  ‣ NN
Software is Important. Why?

Nature of Product Innovation is Shifting

• More than 90% of R&D is related to software according to Ericsson
  – The world’s 5th largest software company
• 70% of all innovation is related to software according to AB Volvo
• 80-90% of all innovation in a car is related to electronics (HW & SW) according to Volvo Cars

https://hbr.org/2015/06/does-hardware-even-matter-anymore

source: J. Bosch,
Speed, Data and Ecosystems: The Future of Software Engineering
“Our civilisation runs on software”
Bjarne Stroustrup, 2008
(inventor of the C++ programming language)

“Software is eating the world”
Marc Andreesen, 2011
(Netscape co-founder)

“Every Company is Now a Software Company”
Satya Nadella, 2019
(CEO Microsoft Corp.)
Software Project Failures

♦ Chaos Report, Standish Group, 2015
  ‣ success: on time, on budget, with satisfactory result

<table>
<thead>
<tr>
<th></th>
<th>2011</th>
<th>2012</th>
<th>2013</th>
<th>2014</th>
<th>2015</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUCCESSFUL</td>
<td>29%</td>
<td>27%</td>
<td>31%</td>
<td>28%</td>
<td>29%</td>
</tr>
<tr>
<td>CHALLENGED</td>
<td>49%</td>
<td>56%</td>
<td>50%</td>
<td>55%</td>
<td>52%</td>
</tr>
<tr>
<td>FAILED</td>
<td>22%</td>
<td>17%</td>
<td>19%</td>
<td>17%</td>
<td>19%</td>
</tr>
</tbody>
</table>

FACEBOOK Engineering:

“… This was the source of yesterday’s outage. During one of these routine maintenance jobs, a command was issued with the intention to assess the availability of global backbone capacity, which unintentionally took down all the connections in our backbone network, effectively disconnecting Facebook data centers globally. Our systems are designed to audit commands like these to prevent mistakes like this, but a **bug in that audit tool** prevented it from properly stopping the command.

This change caused a complete disconnection of our server connections between our data centers and the internet. And that total loss of connection caused a second issue that made things worse. …”

[source: https://engineering.fb.com/2021/10/05/networking-traffic/outage-details/, retrieved October 14, 2021]
New York Times:

“… The investigation was prompted by at least 11 accidents in which Teslas using Autopilot, an assisted-driving system that can steer, accelerate and brake on its own, drove into parked fire trucks, police cars and other emergency vehicles, the safety agency, the National Highway Traffic Safety Administration, disclosed. Those crashes killed one woman and injured 17 people.

…

Tesla’s Autopilot system appears to have difficulty detecting and braking for parked cars generally, including private cars and trucks without flashing lights.

…”

What to Do? Software Processes!

- E.g., V-Model

What? - how?

- Requirements Analysis
- System Design
- Program Design
- Coding
- Unit and Integration Testing
- System Testing
- Acceptance Testing
- Operation and Maintenance

usage

verification & validation

programming
“Program testing can be used to show the presence of bugs, but never to prove their absence.”

E. W. Dijkstra, 1972
Remaining Errors

♦ But…
  › … despite of carefully design, review and testing there are usually 2-3 remaining errors per 1000 lines of code in the program code

♦ Thus…
  › … formal verification of very critical system and code parts
Why is Software so hard?

♦ Complexity
  ‣ medium to big embedded systems (e.g. cars)
    – ~ 500.000 - 2.000.000 lines of code
  ‣ very large software systems (e.g. Windows 10, Airbus A350, Google SW-Ecosystem)
    – > 50 million lines of code

♦ Easy to Change
  ‣ leads to "code-and-fix" paradigm
    – a.k.a. "agile development"

♦ Discrete Artefact
  ‣ the correctness of a program cannot be approximated like in other engineering disciplines (e.g. finite elements, flow calculations) by adding quantitative contingencies

♦ Miscommunicated and Continuously Changing Requirements
  ‣ "When I saw the system for the first time, I knew that it wasn’t what I wanted!“
(Conventional) Software is Text...

... but has (mathematical) structure

control-flow graph

```c
void selectionSort(int arr[], int n)
{
    int i, j, min_idx;

    // One by one move boundary of unsorted subarray
    for (i = 0; i < n - 1; i++)
    {
        // Find the minimum element in unsorted array
        min_idx = i;
        for (j = i + 1; j < n; j++)
        {
            if (arr[j] < arr[min_idx])
                min_idx = j;
        }
        // Swap the found minimum element with the first element
        if (min_idx != i)
            swap(&arr[min_idx], &arr[i]);
    }
}
```
(Conventional) Software is Text...

♦ ... and even logical semantics
  ‣ "software is logic at work"

void selectionSort(int a[], int n);

\[
\forall i \in \mathbb{N}. i \leq n \implies a[i - 1] \leq a[i]
\]

Specification
Model Checking

♦ Model-based Software Engineering and Formal Verification
Verification Method of Model Checking
**Verification Method of Model Checking**

- **Specification**
  - „there shall never be a train and a car in the railroad crossing at the same time“

- **Model Checking**
  - state space search

\[ M \models S \]
Research Areas at our Chair

♦ I. Safety Analysis, Causality, Real-Time Systems and Repair
  ‣ Causality Checking
  ‣ Causality in DNNs
  ‣ Functional Safety of Automotive Systems
  ‣ QuantUM+: Model Based System Engineering, implementation of Causality Checking
  ‣ TarTar: Analysis and Automated Repair of Timed Systems
    – synthesis of repairs using SMT technology

♦ II. Legal Tech
  ‣ logical modeling and analysis of legal artefacts
  ‣ understanding legal contracts using Natural Language Processing

♦ III. Formal Verification and Repair for Machine Learning
  ‣ counterexample computation for DNNs
  ‣ automated repair of DNNs
  ‣ applications in health science

♦ IV. Centre for Human | Data | Science
Research Areas at our Chair

♦ I. Safety Analysis, Causality, Real-Time Systems and Repair
  ‣ Causality Checking
  ‣ Causality in DNNs
  ‣ Functional Safety of Automotive Systems
  ‣ QuantUM+: Model Based System Engineering, implementation of Causality Checking
  ‣ TarTar: Analysis and Automated Repair of Timed Systems
    – synthesis of repairs using SMT technology

♦ II. Legal Tech
  ‣ logical modeling and analysis of legal artefacts
  ‣ understanding legal contracts using Natural Language Processing

♦ III. Formal Verification and Repair for Machine Learning
  ‣ counterexample computation for DNNs
  ‣ automated repair of DNNs
  ‣ applications in health science

♦ IV. Centre for Human | Data | Science
♦ Railroad Crossing

- safety property: $\square \neg (Cc \land Tc)$
- causes for violations?

![Diagram of railroad crossing model]

- $\neg Cc$: Train causes no violation
- $\neg Tc$: Car causes no violation
- $Cc$: Gate causes violation
- $Tc$: Gate causes violation

- $Cc$: Gate is closed
- $Tc$: Gate is open

Causes for violations:
- $Cc$: Gate closed
- $Tc$: Gate open

Diagram elements:
- Ta: Train
- Ti: Car
- Gc: Gate closed
- Go: Gate open
Automated Verification

\[ M \models S \]

- model of the software (transition system, Kripke structure)
- model checking algorithm
- requirement specification (assertions, temporal logic)

• state space search
• symbolic fixed point computation
• SAT/SMT solving

there is never a train in the crossing at the same time when there is a car in the crossing

\[ \neg (Cc \land Tc) \]
Causality Checking

**QuantUM Tool [SPIN 2014, FMICS 2018]**
- support for automated safety analysis
  - e.g., safety evidence according to DO 178C / ISO 26262

- UML/SysML Case Tool
  (IBM Rhapsody, Enterprise Architect, Artisan Studio, Papyrus...)
- QuantUM (Causality Checking)
- Fault Tree Visualization
- SpinJa
- SpinCause
- PRISM
- OSLC
- SysML bdd, stm
- FT
Causality Checking

- Railroad Crossing

  - safety property: $\square \neg (Cc \land Tc)$
  - causal explanation (dynamic causality)
    - causal ordered sequences of events, dynamic fault tree
Computing All Minimal Repairs

- iterative scheme
  - $\mathcal{F}_{i}^{bv}$: bound variation variables forced to 0 in iteration $i$
  - $\hat{\beta}_{i}^{bv}$: bound variation variables set to non-zero value in iteration $i$

\[
\mathcal{F}_{i+1}^{bv} \equiv \mathcal{F}_{i}^{bv} \land \bigwedge_{\hat{\beta}_{i}^{bv}} \hat{\beta}_{i}^{bv} = 0
\]

- example

\[ [1,2] \quad [0,0] \quad [1,1] \quad [1,2] \]
TarTar Tool

- Architecture
  1. UPPAAL model checking on A
  2. UPPAAL-TDT to smtlib2
  3. Z3: compute repair yielding A'
     - quantifier elimination on $(\forall c_i, \delta_j)(T^{bv} \Rightarrow \neg \Phi)$
  4. admissibility check
     1. Ltsmin / Opaal: untimed BAs for A, modified A'
     2. LearnLib: compare $L(A) = L(A')$
Qualitative Experimental Evaluation

♦ Pacemaker Model
  ‣ seeded fault:

<table>
<thead>
<tr>
<th></th>
<th>Original Value</th>
<th>Faulty Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower Rate Interv. (TLRI)</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>Upper Rate Interv. (TURI)</td>
<td>400</td>
<td>1600</td>
</tr>
</tbody>
</table>

♦ property: "between two pacings, maximal TLRI is delayed"

♦ TDT with 9 steps
♦ computed admissible repair:
  - \([\_bv\_clk9\_1 = -600.5] \) TURI – 600.5 = 999.5
♦ SMT problems solved
  - 114 variables, 290 constraints, 1.563 secs.
Research Areas at our Chair

♦ I. Safety Analysis, Causality, Real-Time Systems and Repair
  ‣ Causality Checking
  ‣ Causality in DNNs
  ‣ Functional Safety of Automotive Systems
  ‣ QuantUM+: Model Based System Engineering, implementation of Causality Checking
  ‣ TarTar: Analysis and Automated Repair of Timed Systems
    – synthesis of repairs using SMT technology

♦ II. Legal Tech
  ‣ logical modeling and analysis of legal artefacts
  ‣ understanding legal contracts using Natural Language Processing

♦ III. Formal Verification and Repair for Machine Learning
  ‣ counterexample computation for DNNs
  ‣ automated repair of DNNs
  ‣ applications in health science

♦ IV. Centre for Human | Data | Science
§ 1.1 The Seller Eva hereby sells the shares of Bakery AG with all rights and obligations pertaining thereto (including the dividend right for the current financial year), to the Purchaser Chris who accepts such sale.

1.2 The purchaser pays the purchase price 40,000 € to the seller.

1.3 If the TransferClaim is not performed, the claim.Debtor has the right to withdraw.

1.4 If the PayClaim is not performed, the claim.Debtor has the right to withdraw.

### Parameterized Text Block

<table>
<thead>
<tr>
<th>ID</th>
<th>Block1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>The Seller $seller.Name hereby sells the shares of $shares.Name with all rights and obligations pertaining thereto (including the dividend right for the current financial year), to the Purchaser $purchaser.Name who accepts such sale.</td>
</tr>
</tbody>
</table>

### Objects
- „seller:Person“, „shares:Share“, „transfer:Claim“

### Assignments
- „seller.Name=Eva“, „shares.Name=Bakery AG“, „transfer=$shares.transfer()“

*SPA*

Webinterface:
ContractCheck: Workflow

1. Blocks
2. Object Diagram
3. SMT Tool Lib Code
4. Results

SPA

Diagram showing the workflow:
- Encode
- Formalize
- Z3
- Results
Contract

- property rights, primary / independent and secondary claims

\[ \phi_{SPA} \equiv \phi_{owner} \land \bigwedge_{c \in C} \phi_{c} \land \bigwedge_{c \in C_I} (d_{c} \geq 0 \lor \bigvee_{s \in C(c)} d_{s} \geq 0) \]

- SMT solver can produce model proving satisfiability of \( \phi_{SPA} \)

- preference on primary / independent claims: soft-assert

\[ \phi_{soft} \equiv \bigwedge_{c \in C_I} d_{c} \geq 0 \land \bigwedge_{s \in C(c)} d_{s} = -1 \]

- by solving a partially satisfiable MaxSMT problem the solver provides a model where the execution of consequence claims in minimized
- can be used to show that contract is executable by performing primary claims

- contract execution (partially satisfiable MaxSMT problem)

\[ \land \phi_{soft} \]
Case Study: Pretzel Bakery

Transfer Claim, Analysis I

\[
\Phi_{\text{TransferClaim}} \equiv \phi^B_{\text{owner}} \land \phi_{\text{TransferClaim}} \land d_u \geq 0
\]

\[
\equiv \text{owner(Bakery) = Bank} \land (d_u = -1 \lor (28 \leq d_u \Rightarrow \text{owner(Bakery) = Eva})) \land d_u \geq 0
\]

- functional inconsistency
  - unsat core
- \text{ContractCheck} result

Block 1:
The seller Eva sells the shares of the company Bakery AG to purchaser Chris for 40000 €.

Block 9:
The Bakery AG is transferred by way of security to Bank.
Compensation Claim, Analysis I

- claim limitation before due date
- e.g. analysis of compensation Claim2 (simplified version)
  - $\phi_L \equiv \phi_{SPA} \land \text{Claim2.Limitation} < \text{Claim2.DueDate}$
    $\equiv \phi_{SPA} \land 70 < d_{\text{Warranty}} + 28 + 14$
  - SMT formula unsatisfiable
    - unsat core with $d_{\text{Warranty}} = 29$

![Diagram of Pretzel Bakery case study](image-url)
Research Areas at our Chair

♦ I. Safety Analysis, Causality, Real-Time Systems and Repair
  ‣ Causality Checking
  ‣ Causality in DNNs
  ‣ Functional Safety of Automotive Systems
  ‣ QuantUM+: Model Based System Engineering, implementation of Causality Checking
  ‣ TarTar: Analysis and Automated Repair of Timed Systems
    – synthesis of repairs using SMT technology

♦ II. Legal Tech
  ‣ logical modeling and analysis of legal artefacts
  ‣ understanding legal contracts using Natural Language Processing

♦ III. Formal Verification and Repair for Machine Learning
  ‣ counterexample computation for DNNs
  ‣ automated repair of DNNs
  ‣ applications in health science

♦ IV. Centre for Human | Data | Science
Formal Verification for Machine Learning

- Deep neural networks are used in autonomous driving, aircraft, and medical diagnosis
- These are safety-critical domains
  - We have to ensure that DNNs behave as expected
  - Challenge: Inner workings of DNNs are opaque to humans

Henzinger, Lukina and Schilling, 2020
Irfan et al., 2020
End-to-end Autonomous Driving (Nvidia)

Research Project II: Formal Verification of Deep Neural Networks

Deep Neural Networks (DNNs) / Convolutional Neural Networks

- issues
  - non-linear, statistical decision making
    
    $f(\Sigma_i x_i w_i + b)$

- absence of a easily interpretable logical structure
- very large state spaces ("the curse of dimensionality")

- specifications?
  - $M \models S$?

- approaches
  - linear arithmetic decision procedures (SMT solving)
  - approximative techniques
  - precision / scalability?
How to Verify and Repair Deep Neural Networks

♦ Verification using Satisfiability Solvers (SMT)
  ‣ Encode the network as a logical formula and solve

♦ Verification using Optimization
  ‣ Encode Verification as an optimization problem
  ‣ Use optimization techniques that can find global optima
  ‣ Many techniques from mixed-integer linear programming
    ‒ Linear Relaxations
    ‒ Branch and Bound
    ‒ Recently: Cutting Planes

Linear Relaxation of a ReLU neuron, following Xu et al. 2021 (ICLR).
SpecRepair

♦ Overview

Counter-example generation

- Specification $\varphi(x)$
- DNN $N$

Counter-example search

- no
- Verification

- yes
- Specification satisfied?

- yes
- Output repaired DNN $N$

- no

Repair

- Retrain network
- Counter-examples $x^c_j$
- Penalty weights
- Data set $(X, Y)$ (Optional)
ACAS Xu

- repair of property $\varphi_2$ "clear-of-conflict is never the maximum output"
Research Areas at our Chair

♦ I. Safety Analysis, Causality, Real-Time Systems and Repair
  ‣ Causality Checking
  ‣ Causality in DNNs
  ‣ Functional Safety of Automotive Systems
  ‣ QuantUM+: Model Based System Engineering, implementation of Causality Checking
  ‣ TarTar: Analysis and Automated Repair of Timed Systems
    – synthesis of repairs using SMT technology

♦ II. Legal Tech
  ‣ logical modeling and analysis of legal artefacts
  ‣ understanding legal contracts using Natural Language Processing

♦ III. Formal Verification and Repair for Machine Learning
  ‣ counterexample computation for DNNs
  ‣ automated repair of DNNs
  ‣ applications in health science

♦ IV. Centre for Human | Data | Science
Projects and Theses

◆ What we offer:

✦ usually every semester, proposal for approx. 15 projects and theses (Bachelor / Master)
  – often closely related to research projects
  – own proposals welcome!
  – project leads up to thesis, introductory / literature study phase

✦ seminars

✦ Hiwi (student assistant) opportunities

◆ Presentation of Project/Thesis Topics

✦ following this presentation

✦ will be posted on
Lectures / Courses Offered

♦ **Software-Engineering** (L)
  ‣ 3rd/4th semester Bachelor, Summer Semester

♦ **Lectures for Advanced Bachelor and Master Students**
  ‣ Verification of Software and Systems (L)
  ‣ *Advanced Verification of Software and Systems (L)*
  ‣ Logic in Computer and Software Science (L)
  ‣ Decision Procedures for Software Verification (DS)
  ‣ Verification of Cyber-Physical Systems (DS)
  ‣ System Specification using TLA+ (DS)
  ‣ Concurrent, Distributed and Cloud Computing (DS)
  ‣ Functional Safety in Embedded Systems (DS)
  ‣ Software Testing (DS)
  ‣ Requirements Engineering (DS)
  ‣ Software Architecture (DS)
  ‣ various seminars
Questions & Discussion