Panel #2

VENICE
FALL 2024

SoftNet 2024 Congress

Theme:
Software-Now – Developing, Simulation, and Validation Challenges

Panel #2

VENICE
FALL 2024

Moderator

Prof. Dr. Hans-Werner Sehring, NORDAKADEMIE gAG, Germany

Panelists

Prof. Dr. Simona Vasilache, University of Tsukuba, Japan

Prof. Dr. Radek Kočí, Brno University of Technology, Czech Republic

Prof. Dr. Luigi Lavazza, Universita` degli Studi dell'Insubria, Italy

Prof. Dr. Carlo Simon, Hochschule Worms, Deutschland

Jos van Rooijen, Huis voor Software Kwaliteit, Nederland

Dr. Hayk Aslanyan, CAST, RAU, Armenia

Panel Moderator

VENICE
FALL 2024

Before we hear the panelists' positions, let me give you some background on myself,

… and my opinion on the topic.

Hans-Werner Sehring
NORDAKADEMIE

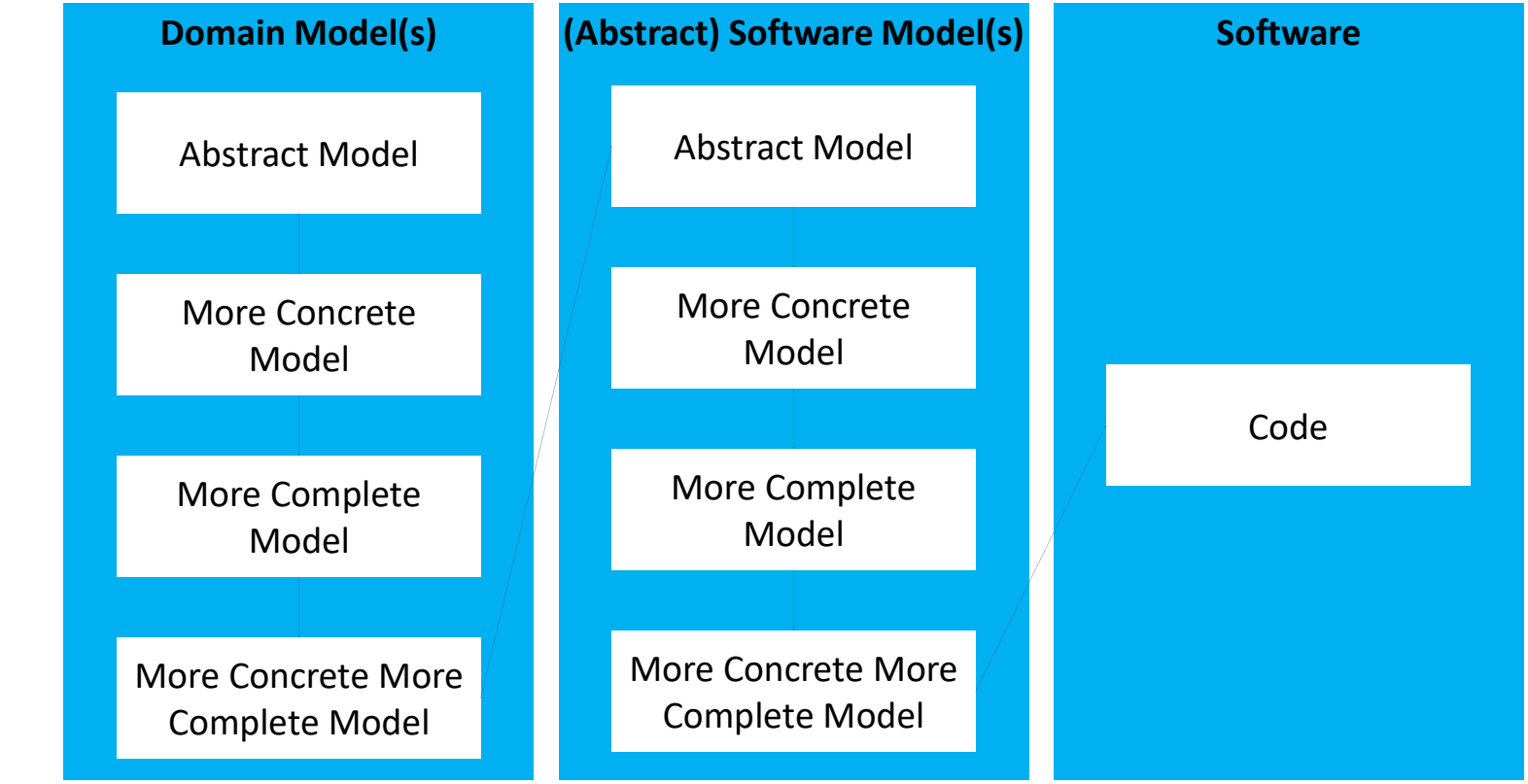
- **Me**
  - Professor for Software Engineering at NORDAKADEMIE "Hochschule der Wirtschaft" (University of the economy)
  - Scientific and practical background
- **Research interest**
  - Model-driven Software Engineering
  - Domain modeling
  - Software specification
  - Programming language specification

- **For the topic of the panel**
  - In practice, software is tested as part of a quality assurance process.
  - We all know that in fact we should proof correctness.
  - Idea from formal program semantics: if proof is constructive, then it is a software generator.

Panel Moderator's Position

VENICE
FALL 2024

Idealized MDSE Captures Three Domains: Subject Domain, Software Specification, and Code



Hans-Werner Sehring
NORDAKADEMIE

Panel Moderator's Position

VENICE
FALL 2024

A perspective von Model-Driven Software Engineering:
When code generation is proofably correct, quality is granted?

Hans-Werner Sehring
NORDAKADEMIE

Subject domain semantics

- Was the **problem** modeled correctly?
- Are all **requirements** specified? Are all **constraints** considered?
- Plus: with Generative AI there is a trend to go back to **prosaic descriptions**.

Software design

- Does the software **specification** address all requirements and constraints?
- Is it practical? With most projects being agile, there is direct **feedback** from implementations.

Code

- Are software generators working correctly? Including code catering for non-functional requirements?
- How about deployment, changing environments, evolution, etc.?

Therefore, even when software was built correctly through a correct generation process…
… there still can be domain modeling and software design flaws. How are these tested (in isolation)?

Panelist Position

VENICE
FALL 2024

Software-Now - Developing, Simulation, and Validation Challenges

Carlo Simon
Univ. Worms, Germany

Product

- Traceable requirements
- Definition of Done
- Separate testing from development
- Product Backlog vs. Sprint Backlog

People

- Skilled labor shortage -> Career changer
- Stay up-to-date
- Use AI generated software
- “Low code”-worker

Domain

- Production, Logistics
- Maintainable vs. hard constraints
- Users as part of the team
- Development of AI applications (LLM prompts)

Panelist Position

VENICE
FALL 2024

- **Software-Now - Developing, Simulation, and Validation Challenges**
- **After so many years, the development process is still a problem**
  - **Even with the available technology, setting up an efficient, effective and evolvable development process is huge problem**
  - **How to write requirements**
    - **Requirements "culture"**
    - **Impedance mismatch among business analysts, GUI designers, developers, testers**
  - **How to structure the development organization**
    - **By product vs by competence**
    - **By contract vs by (reusable) components**
    - **…**
  - **What process model?**
    - **Agile everybody?**
  - **What tools**
    - **Scouting**
    - **Configuration / customization**
    - **Lock-in**

Luigi Lavazza
Univ. Insubria

Panelist Position

VENICE
FALL 2024

Luigi Lavazza
Univ. Insubria

- **Software-Now - Developing, Simulation, and Validation Challenges**
- **After so many years, the development process is still a problem**
  - **How designers (mock-ups, prototypes) fit in the requirements definition process**
    - Clash with front-end developers
    - Suboptimal technical choices
    - Cost issues
  - **Organization Structure**
    - To effectively support projects
      - The usual dilemma: BUs vs competence centers
    - To support transition
      - E.g., monolithic to microservice-based

Panelist Position

VENICE
FALL 2024

Luigi Lavazza
Univ. Insubria

- **Software-Now - Developing, Simulation, and Validation Challenges**
- **After so many years, the development process is still a problem**
  - **Tools**
    - **What tools are available?**
    - **How do they fit in the process?**
      - **As-is**
      - **To-be (hopefully, some improvement is envisioned)**
    - **How much do they cost? (also in terms of learning curves)**
    - **How easily can we switch to different tools, if needed?**

Panelist Position

VENICE
FALL 2024

Simona Vasilache
University of
Tsukuba, Japan

- **Software (engineering) education: must keep up with *all* developments**
- **Students' interests**
  - **Front-end / back-end development**
  - **Machine learning, data engineering, AI etc.**
- **Q: Software developer OR Software engineer ?**
- **Agile development – highly popular with students**
  - **Attractiveness of startups *("The Lean Startup" – Eric Ries)***

Panelist Position

VENICE
FALL 2024

Simona Vasilache
University of
Tsukuba, Japan

- **Testing: costly and complex phase in the software development process**
- **Challenge: educating students (→testers) with the right skillset**
  - **Software testing concepts included in coursework OR**
  - **Full dedicated courses**
  - **Puzzling question in class:**
    "What is the difference between verification and validation?"
- **Problems**
  - **Testing and maintenance: least glamorous activities**
  - **Students (and everyone else?!) perceive testing as *dull, difficult, non-creative***

Panelist Position

VENICE
FALL 2024

Simona Vasilache
University of
Tsukuba, Japan

- Various approaches to <u>software testing education</u>
  - Increasing motivation for learning about testing (teamwork?!)
  - Software testing concepts in introductory programming courses
  - Real projects that use industry-tested tools
  - Gamification
    - Fun and/or easy-to-implement games in the classroom
- "Test-driven development" (TDD)
  - Strong opinions, both for and against!
- Future: using AI
  - Why bother with learning about testing?! ☺

Panelist Position

VENICE
FALL 2024

Jos van Rooijen
Huis voor Software Kwaliteit

- **Consultant software testing / Quality Supervision**

- Software testing is still really traditional
- Innovation and research is below par
- The (so called) innovation concentrates:
  - How to apply test in a new development method
  - Test tooling
- Hardly none testing techniques to beat future challenges we have to face, like:
  - Self driving cars
  - Code development by hand of AI (how to prove the code)
  - Dependencies of medical devices
  - Smart devices connected to everything and everywhere
  - Etc.

Panelist Position; The context

VENICE
FALL 2024

Threads

Development inside quality engineering

Chances

The context

Future of software testing

Social development

?

Development of Information Technology

Panelist Position

Some challenges per perspective:

Developments Quality Engineering:

- Increasing dynamics. Development of information systems is never finished. So testing is also never finished!
- Increasing complexity
- Bugs appears on different levels. Configuration, integration or parametrisation

Threads:

- Low chance, high impact
- Aging
- Complexity
- Self learing information systems; we don’t know any more how the information system works
- Lack of cooperation between the industry and academia

What are the measures we have to take?

- What kind of techniques / approaches we have to develop?
- Is there something available?
- Traditional test approaches are not applicable anymore

Panelist Position

VENICE
FALL 2024

- **Does AI influence the way software develops?**
  - Code generation
  - Test Generation
  - What about requirements or design activities?
- **Does AI influence developers?**
  - Will there be a need for as many (low-level) programmers?
  - In general, can AI replace junior positions?
  - If so, where will developers get experience for senior positions?
- **Challenges**
  - It will be necessary to prepare (and teach software engineers) for changes in development processes
  - Less emphasis on programming
  - More emphasis on analysis and design
  - Need to be able to interact appropriately with AI and be able to evaluate AI results
  - Combination code generation and formal verification (or other methods) can reduce the need for programming while maintaining confidence in the code

Radek Kočí,
Brno University of Technology

Panelist Position

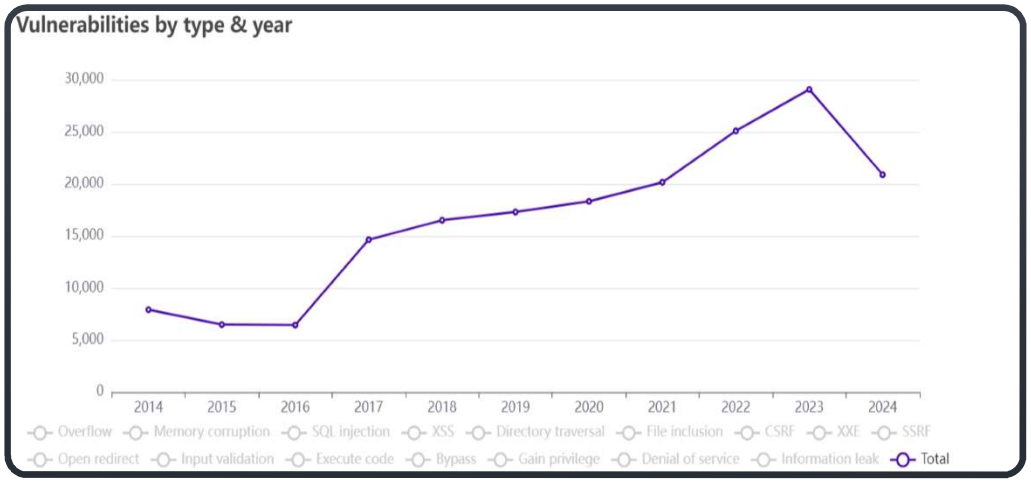
VENICE
FALL 2024

Hayk Aslanyan
CAST, Armenia

- **DEFECTS IN PROGRAMS**
  - **On average, a developer creates 70 bugs per 1000 lines of code**
  - **15 bugs per 1,000 lines of code find their way to the customers**
  - **Fixing a bug takes 30 times longer than writing a line of code**
  - **75% of a developer's time is spent on debugging**
  - **In the US alone, ~$113B is spent annually on identifying & fixing product defects**
  - **…….**

Panelist Position

VENICE
FALL 2024



https://www.cvedetails.com/

Hayk Aslanyan
CAST, Armenia

CVE cam be duplicated multiple times.

Examples:

- ***OpenSSL HeartBleed*** (leak of encrypted information), CVE-2014-0160
- ***Equifax Data Breach*** (147m. personal data leak), known CVE-2017-5638 in Apache Struts

Panelist Position

VENICE
FALL 2024

SECURITY MUST BE CONSIDERED IN ALL STAGES OF SOFTWARE DEVELOPMENT:

- **REQUIREMENTS GATHERING STAGE.** PREPARE AN **APPLICATION RISK PROFILE**. THE DOCUMENT DESCRIBES POSSIBLE ENTRY POINTS FOR ATTACKERS AND CATEGORIZES SECURITY RISKS BY THE SEVERITY LEVEL, INCLUDING THEIR IMPACT AND LIKELIHOOD.
- **SOFTWARE DESIGN STAGE**. **THREAT MODELING** WHEN HIGH-LEVEL SOFTWARE ARCHITECTURE IS DESIGNED, AND POSSIBLE DATA FLOWS AND DATA ENTRY POINTS ARE ESTABLISHED. IT INCLUDES:
  - *DECOMPOSING THE APPLICATION ARCHITECTURE INTO FUNCTIONAL COMPONENTS*
  - *DETERMINING THREATS TO EACH OF THE COMPONENTS*
  - *CATEGORIZATION AND PRIORITIZATION*
  - *PLANNING COUNTERMEASURES FOR POSSIBLE ATTACKS*
- **SOFTWARE DEVELOPMENT STAGE.**
  - *SECURE CODING PRACTICES*
  - ***STATIC ANALYSIS***
  - ***DYNAMIC ANALYSIS***
  - *REGULAR PEER REVIEW*
- **SOFTWARE DEPLOYMENT AND SUPPORT STAGE.**
  - *PENETRATION TESTING*
  - *CREATING AN INCIDENT RESPONSE PROCEDURE*
  - *SETTING APPLICATION SECURITY MONITORING (MANUAL AND AUTOMATED)*
  - *SUBMITTING YOUR APPLICATION FOR EXTERNAL VALIDATION*
  - *ESTABLISHING A FEEDBACK PROCESS AND TOOLS FOR USERS (TO REPORT VULNERABILITIES)*

Hayk Aslanyan
CAST, Armenia

STAGE IS YOURS!

Theme:
Software-Now – Developing, Simulation, and Validation Challenges