

ICSEA 2015

Monday, November 16, 2016
Barcelona, Spain

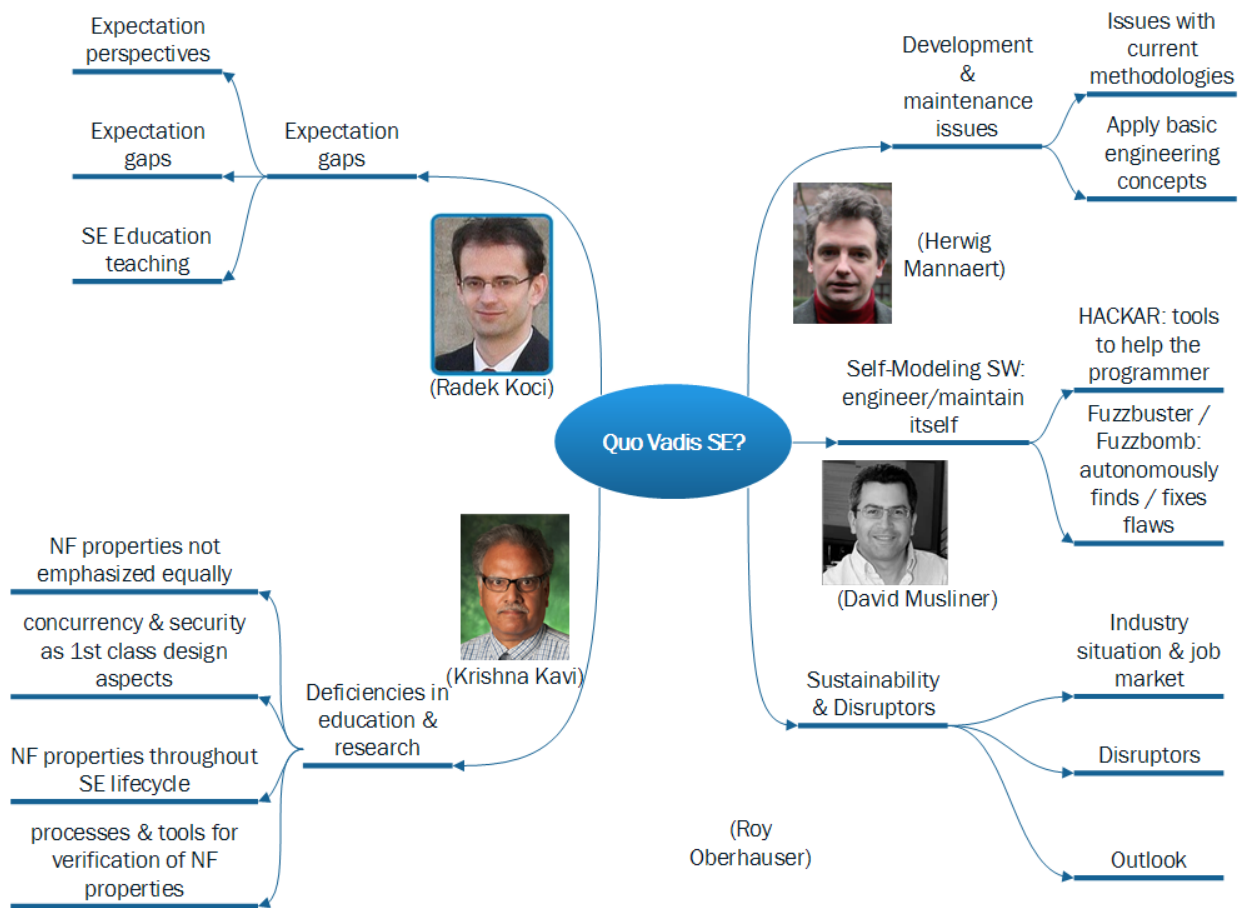
Panel: Quo Vadis Software Engineering?

Moderator

Roy Oberhauser, Aalen University, Germany

Panelists

Kavi Krishna, University of North Texas, USA
Radek Koci, Brno University of Technology, Czech Republic
David Musliner, Smart Information Flow Technologies (SIFT), USA
Herwig Mannaert, University of Antwerp, Belgium



ICSEA 2015 Panel discussion: Quo Vadis Software Engineering?

SUSTAINABILITY & DISRUPTORS

Roy Oberhauser
Aalen University
Germany

SE Quality and Maintenance Problem?

- U.S. software industry revenue 2012: **\$425 billion**
- Cost of bugs to the U.S. economy: **\$60 billion** annually
- Cost and criticality of software to society
- Consider code volume and typical defect rates
- Bugs/vulnerabilities have increasing value
 - Greater *usage/reliance* on software systems
 - More *data* behind any single breach
 - *Misuse* market for discovered defects
 - Widespread *reuse*/dispersement of (defective) code
 - Huge dependency chains (e.g., Heartbleed 1/2/...)

Correction work costs pale in relation to indirect costs and risks of a bug

SE Employment Problem?

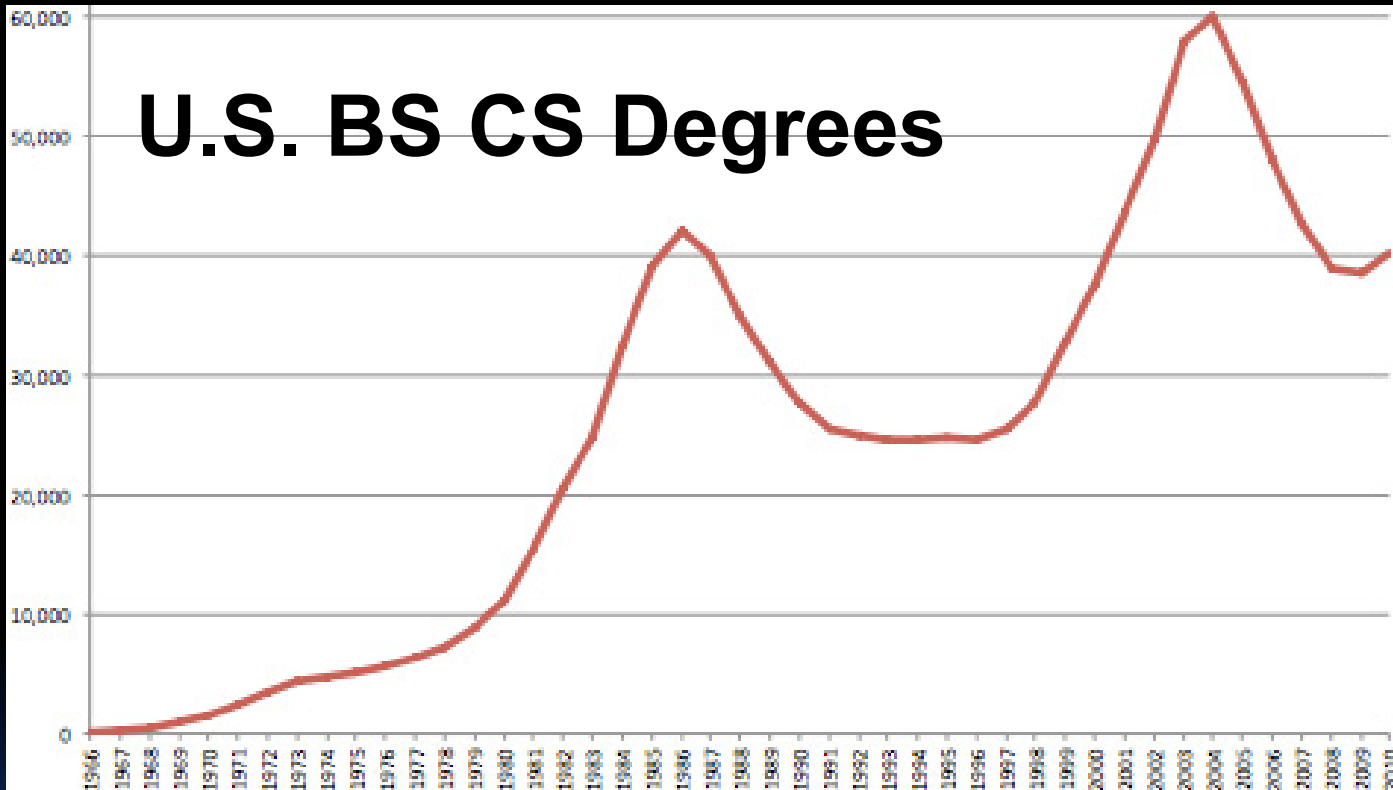
- What are SW engineers doing?
- U.S. SW industry employment distribution:

Year	Development Personnel	Maintenance Personnel	Total Personnel	Maintenance Percent
1950	1,000	100	1,100	9.09%
1955	2,500	250	2,750	9.09%
1960	20,000	2,000	22,000	9.09%
1965	50,000	10,000	60,000	16.67%
1970	125,000	25,000	150,000	16.67%
1975	350,000	75,000	425,000	17.65%
1980	600,000	300,000	900,000	33.33%
1985	750,000	500,000	1,250,000	40.00%
1990	900,000	800,000	1,700,000	47.06%
1995	1,000,000	1,100,000	2,100,000	52.38%
2000	750,000	2,000,000	2,750,000	72.73%
2005	775,000	2,500,000	3,275,000	76.34%
2010	800,000	3,000,000	3,800,000	78.95%
2015	1,000,000	3,500,000	4,500,000	77.78%
2020	1,100,000	3,750,000	4,850,000	77.32%
2025	1,250,000	4,250,000	5,500,000	77.27%

[Casper Jones 2006: The Economics of Software Maintenance in the Twenty First Century]

Let's assume 75% doing maintenance!

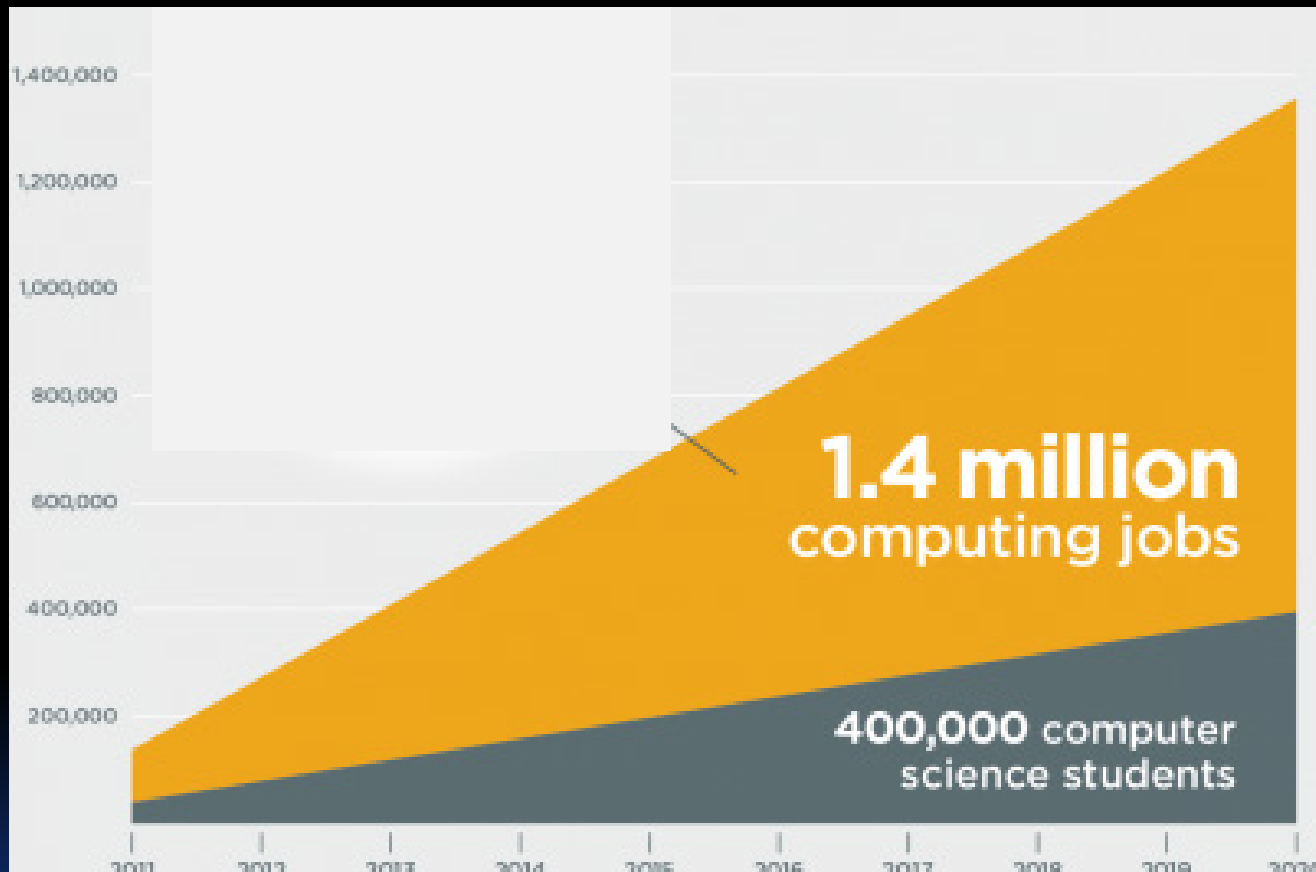
SW Engineer Sustainability Problem?



[adapted from Ed Lazowska, University of Washington]

In 2011 we are back to level seen in 1986.
And unchanged at 1% of 23 year olds.

SW Engineer Sustainability Problem?



Hadi Partov Bureau of Labor Statistics

<http://www.geekwire.com/2014/analysis-examining-computer-science-education-explosion/>

Will/can U.S. even produce 400K grads in 2020?

SE Disruptors? Reuse & Integrators

- Increasing *reuse* of software
- SW engineers primarily becoming “system integrators”?
 - Building business value fast - not from scratch
 - Reuse via frameworks mostly (re)configuring & glueing?
 - Less “expertise” and specialty, more generalists?
 - Software market development:
craft – chaos – segregation – consolidation - saturation?
 - Apps in Stores: 1.6M Google, 1.5M Apple.
Market saturation reached? Do we need a billion apps?
 - >14K Web APIs
 - Users also integrating: via IFTTT

Possible SE Disruptors? The rise of Autonomic properties Self-X properties

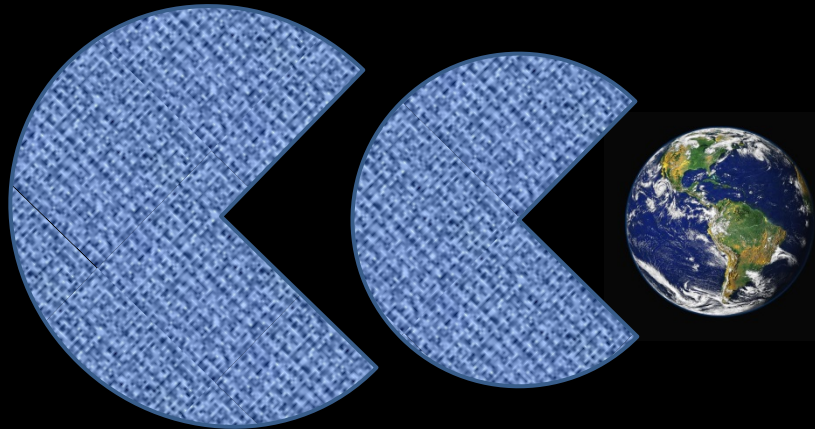
- Self-configuration
- Self-healing
- Self-optimization
- Self-protection
- Self-maintenance
- Self-...
- Automated software engineering
- Natural language programming
- Technically adept society and demographics

SE Disruptor?

Self-programming becomes viable

- AI algorithms plus massive data
 - Data on how we have solved programming tasks
- Creativity: how much of programming today is actually novel ideas and talent?
 - 80% of software no brain work [Ivar Jacobson 2007]
 - Patterns: codify generalized solutions to problems
 - Most programming is manual drudgework
 - No tricky code wanted! (for maintenance reasons)
- If end-users give AI their goals, what is hard?
 - E.g., declarative approaches...

Software is eating the world... [MarcAndreessen]



Will software
eat itself?

Recursive software self-improvement?
A technological singularity for the SE domain?

May software engineers
automate themselves out of a job?

The Age after the SE Disruptors...

- Intentionality and our systems
 - Dark stakeholders will remain
 - Automation of misuse and “control”: by/for whom?
- “Almost all grave software problems can be traced to conceptual mistakes made before programming started”
 - Prof. Jackson of MIT in Scientific American June 2006
 - So perhaps the focus for SW engineers lies beyond the programming
- There will always be room for creative SE... just perhaps not so many jobs openings?
- How will/should this automation impact how we educate the next generation of software engineers?

Quo Vadis Software Engineering?

Krishna M. Kavi

Professor and Director

NSF Net-centric and Cloud Software and Systems

Industry/University Cooperative Research Center

University of North Texas

Where am I coming from?

Unde veni et quo vadis?

Ex quo venio?

Nearly two decades ago dabbled in Formal Methods

Concurrency models using Petri-nets and dataflow graphs

Real-time software systems

Served as a department chair and was involved in ABET curricular issues

Talked with colleagues who teach SE type courses

More recently, my interests (from software engineering viewpoint) are

non-functional properties

Security

Performance Engineering

non Quo Vadis Software Engineering?

What is lacking in SE, particularly in terms of education but also in terms of research.

- We are not emphasizing non-functional properties at the same level as we do with functional properties
- We are not emphasizing concurrency and security as first-class design aspects
- We need to include non-functional properties throughout the SE life cycle
- We need to develop processes and tools that help with verification of non functional properties

Consider the latest ACM/IEEE recommendation for Software Engineering education

ACM/IEEE Recommendations

Definition: Software engineering is the discipline concerned with the application of theory, knowledge, and practice to *effectively and efficiently* build reliable software systems that satisfy the requirements of customers and users.

What do these terms - effectively and efficiently - mean?

Should they include non-functional requirements?

Should they include the use of concurrency for efficiency?

SE/Software Processes


ACM/IEEE Recommendations

Topics:

[Core-Tier1]

- Systems level considerations, i.e., the interaction of software with its intended environment
- Introduction to software process models (e.g., waterfall, incremental, agile)
Activities within software lifecycles
- Programming in the large vs. individual programming

Just a mention of
“environment”




[Core-Tier2]

- Evaluation of software process models

[Elective]

- Software quality concepts
- Process improvement
- Software process capability maturity models
- Software process measurement

Should quality include
how well non-functional
requirements are met?



SE/Requirements Engineering

ACM/IEEE Recommendations

Topics

[Core-Tier1]

- Describing functional requirements using, for example, use cases or users stories
- Properties of requirements including consistency, validity, completeness, and feasibility

[Core-Tier2]

How to elicit non-functional requirements

- Software requirements elicitation
- Describing system data using, for example, class diagrams or entity-relationship diagrams
- Non-functional requirements and their relationship to software quality (cross-reference IAS/Secure Software Engineering)
- Evaluation and use of requirements specifications

Non-functional requirements may impact the design itself

[Elective]

How to analyze non-functional requirements?

- Requirements analysis modeling techniques
- Acceptability of certainty / uncertainty considerations regarding software / system behavior
- Prototyping
- Basic concepts of formal requirements specification
- Requirements specification
- Requirements validation

SE/Software Design

ACM/IEEE Recommendations

Topics

[Core-Tier1]

Should architecture design account for “concurrency” and other non-functional properties?

- System design principles: **levels of abstraction** (architectural design and detailed design), separation of concerns, information hiding, coupling and cohesion, re-use of standard structures
- Design Paradigms such as structured design (top-down functional decomposition), object-oriented analysis and design, event driven design, component-level design, data-structured centered, aspect oriented, function oriented, service oriented
- Structural and behavioral models of software designs
- Design patterns

Performance design patterns
Concurrency patterns
Security patterns?

[Core-Tier2]

- Relationships between requirements and designs: transformation of models, design of contracts, Invariants
- Software architecture concepts and standard architectures (e.g. client-server, n-layer, transform centered, pipes-and-filters)
- Refactoring designs using design patterns
- The use of components in design: component selection, design, adaptation and assembly of components, components and patterns, components and objects (for example, building a GUI using a standard widget set)

Where Should Software Engineering Be Going

Quo ire Software Engineering?

Consideration of Security and Concurrency as a fundamental design concepts

Emphasize Non-Functional Properties along with Functional

Requirements elicitation

Architecture design

Design Patterns

Life cycle

test generation

New tools

Component Engineering to meet non-functional requirements

Where Should Software Engineering Be Going

Quo ire Software Engineering?

Research

Concurrency modeling

- not just swim lanes
- my need to integrate Petri nets with UML and other modeling systems

Concurrency testing

- replay synchronization orders
- detecting race conditions, deadlocks and livelocks
- load balancing, etc

Architecture

- how to describe non-functional aspects in the architecture

Design patterns

- concurrency patterns
- Performance patters and anti-patterns
- security patters

Life cycle

- not sure if we need to change any phase in a life cycle
- may be software maintenance – and monitoring?

Tools

- to capture non-functional requirements
- aid in design, testing ..

Where Should Software Engineering Be Going

Quo ire Software Engineering?

Architecture:

Which architecture is better to handle concurrency, security, reliability..

SoA?

Layered?

Client-Server

.....

Where Should Software Engineering Be Going

Quo ire Software Engineering?

Life Cycle

In terms of security

May need to focus on *monitoring* after deployment

patches and complete upgrades

Can this be viewed as a part of maintenance?

May be similar phases for other non-functional properties

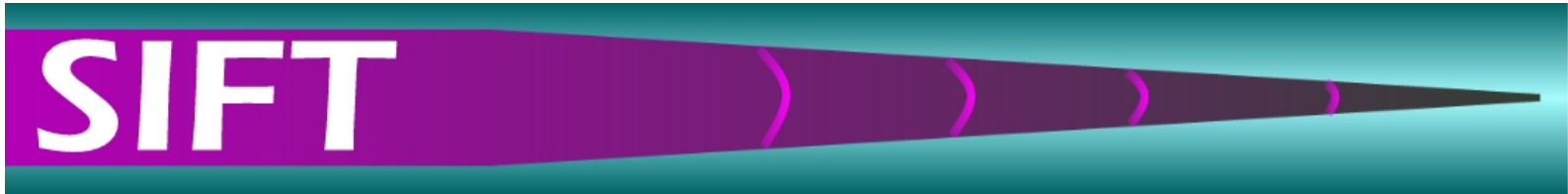
New processors may require changes to deployed software

redesign, or adapt

Where Should Software Engineering Be Going

Quo ire Software Engineering?

- Software complexity is increasing
- Not only functional complexity but also in terms of non-functional requirements
- Need to take into account and manage non-functional requirements throughout the software engineering processes
- Component engineering must address non-functional requirements



Smart Information Flow Technologies

Software Engineering Panel

Dr. David J. Musliner

musliner@sift.net

(612) 325-9314

For other staff and projects, please see

www.sift.net

Who are we?

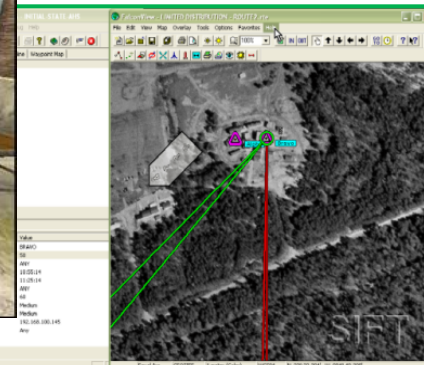
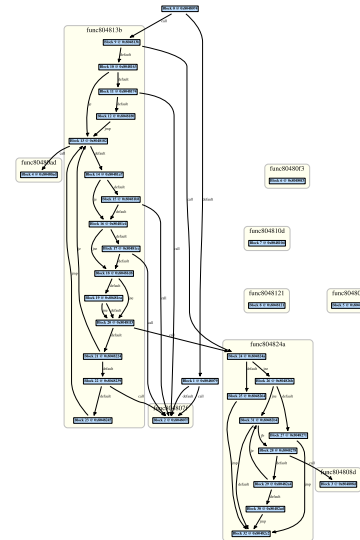
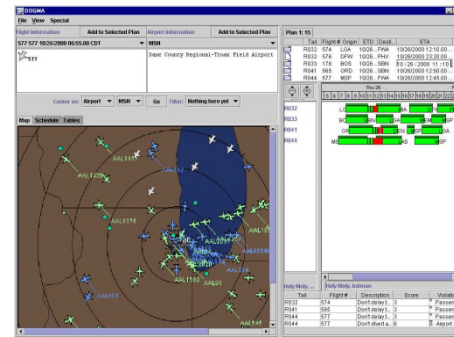
- 35 person small business based in Minneapolis, branches in Boston, D.C., Dallas, San Diego.
- 15 years in business.
- \$7M+ per year in revenues.
- 30 advanced degrees (Computer Science, Psychology, Control).
- 150+ years combined industrial R&D experience.
- Customers include DARPA, NASA, NIST, AFRL, ARL, ONR, OSD, AFOSR, Lockheed, BBN, BAE Systems...
- Research spanning computer science, human centered systems, and beyond.



Downtown Minneapolis

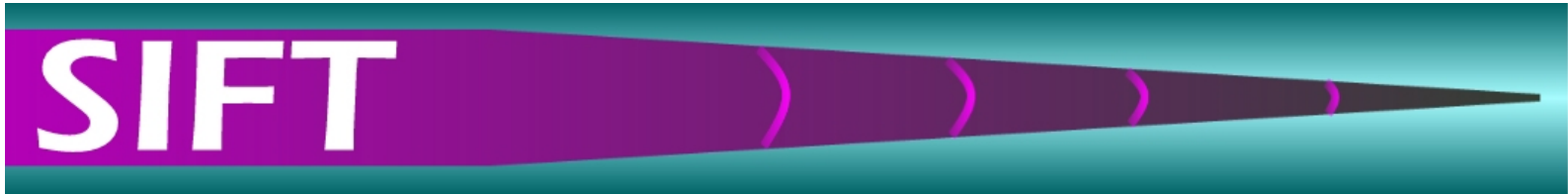


Lexington, MA



SIFT Technology Thrust Areas

- Cyber security.
- Autonomous systems.
- Formal verification and statistical model checking.
- Natural language and information extraction.
- Human cognition and performance.
- Etiquette and socially-aware systems.
- Dynamic information management & presentation.

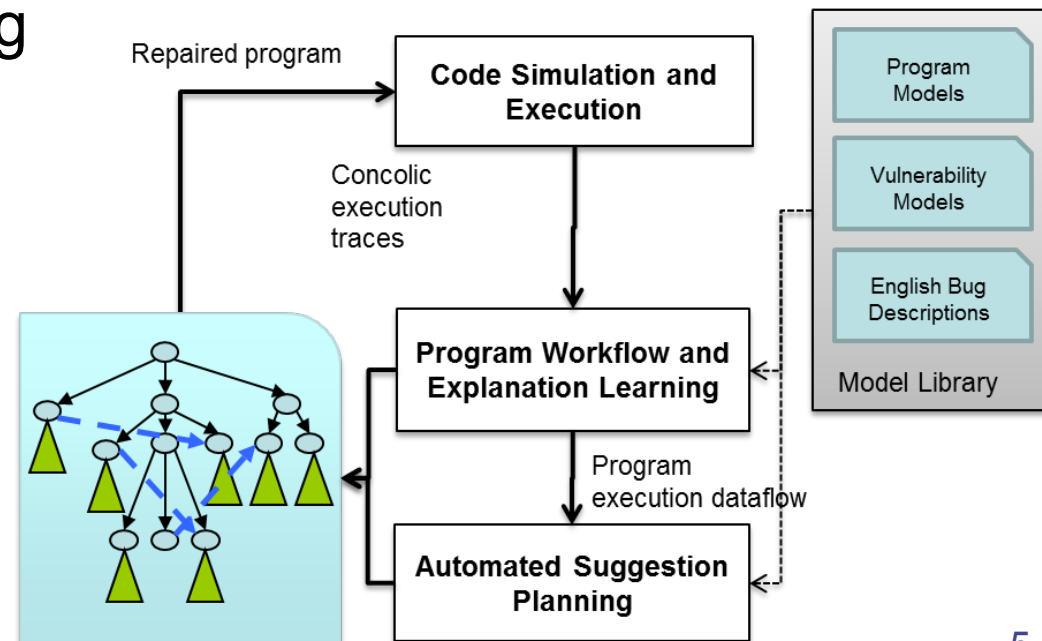


Cyber-Security

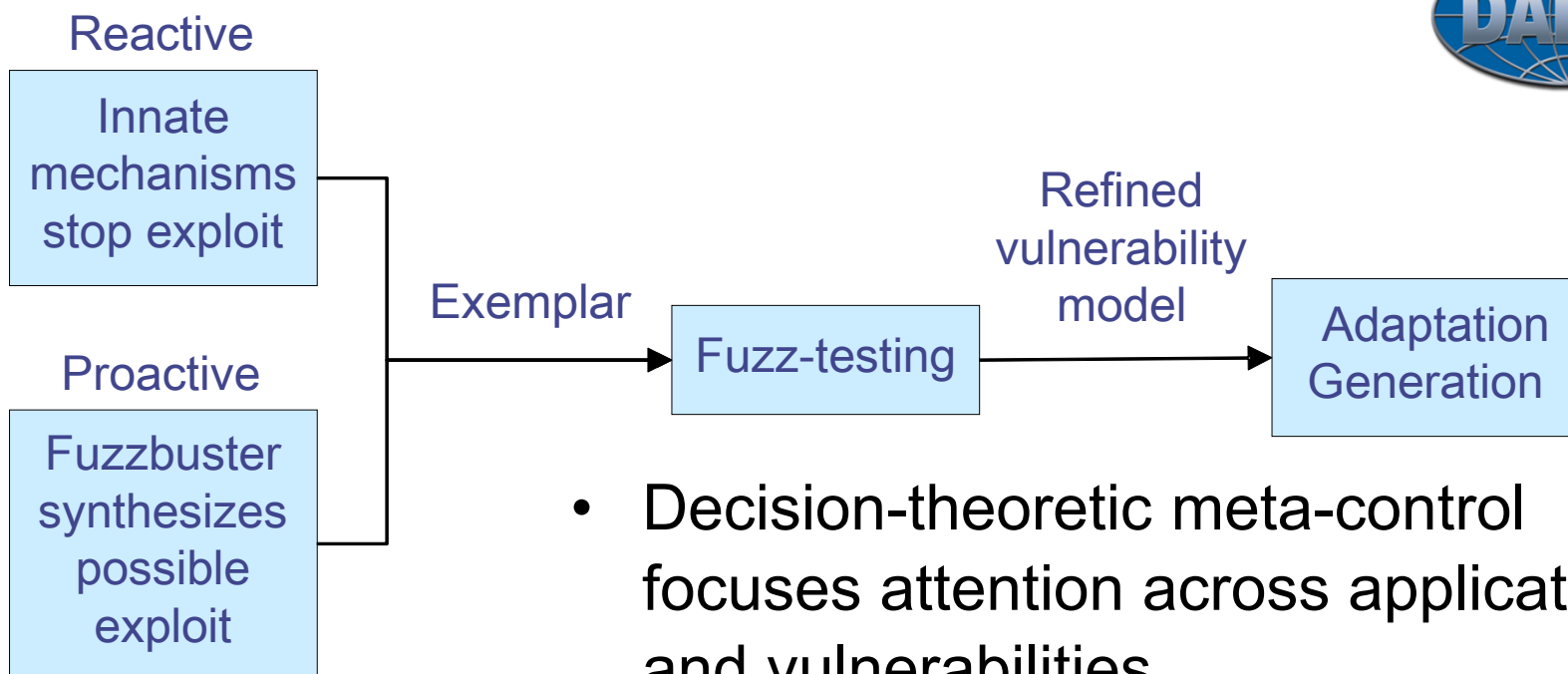
- Tools for developing safe software.
- Fully autonomous vulnerability detection and mitigation, for deployed software.
- Attack detection and plan recognition.
- Mission-aware cloud adaptation and defense.
- Agents for cyberwar simulation and traffic generation.

HACKAR: Helpful Advice and Code Knowledge for Attack Resistance

- Detects and diagnoses vulnerabilities in programs **at development-time**.
- Generates models of software as hierarchical workflows to find the causes of vulnerabilities, explain problems, and suggest code alternatives.
- Eclipse plug-in collaborates with programmer.
- Speeds safe programming 2X, fault repair 5X.
- Vulnerability rate reduced 80%.
- ONR funded.



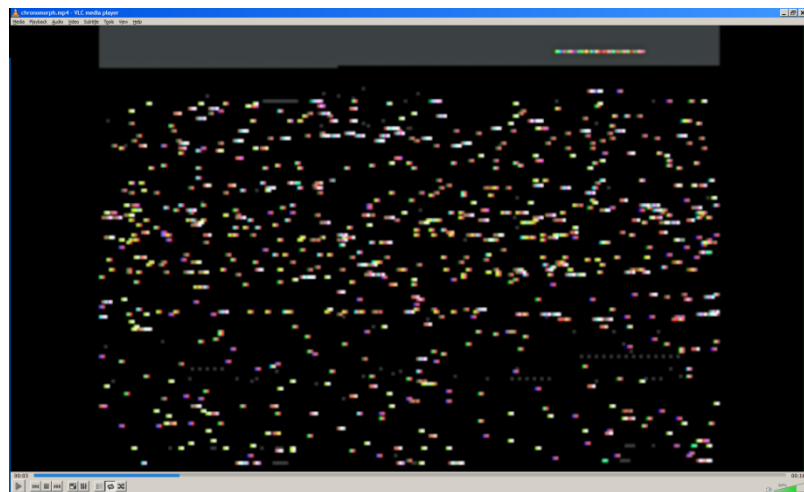
- Reactively and proactively use fuzz-testing and other tools to detect and characterize software vulnerabilities.
- Adaptively repair or shield vulnerabilities, preventing future exploitation.
- Fuzzbuster has automatically found and shielded dozens of vulnerabilities.



- Decision-theoretic meta-control focuses attention across applications and vulnerabilities.
- DARPA CRASH program.

- Symbolic analysis of binaries to identify vulnerabilities and craft exploits.
- Binary rewriting methods to prevent exploitation.
- Massively distributed.
- Fully autonomous.

- Chronomorphic binaries.
 - Rewrites and moves gadgets repeatedly, at runtime.
 - Defeats ROP and BR0P attacks.
 - Proof of concept works directly on binary.



Attack Detection and Intent Recognition

- Scyllarus event correlator accumulates low-level cyber security events and alerts into higher-level incident reports.
 - Reduces alert volume 1000X+.
 - Computes criticality and severity.
 - DARPA Integrated Learning, Scalable Network Monitoring programs.
 - **Transitioned to ISC8/CyberAdapt commercial network appliance.**
- YAPPR probabilistic plan recognition system analyzes event traces to compute likely attacker plans.
 - DARPA Self-Regenerative Systems, Integrated Learning, Mission-oriented Resilient Clouds programs.





Quo Vadis Software Engineering ?

Prof. dr. Herwig Mannaert

Universiteit Antwerpen



Software Engineering

- Huge achievements I will not talk about ...
- Many new evolutions I will not talk about ...
- Many new technologies I will not talk about ...

- I will talk a bit about a crucial challenge and some basic needs/concepts to address this



Software Challenge (one of ...)

The Law of Increasing Complexity **Manny Lehman**

“As an evolving program is continually changed, its complexity, reflecting deteriorating structure, increases unless work is done to maintain or reduce it.”

Proceedings of the IEEE, vol. 68, nr. 9, september 1980, pp. 1068.

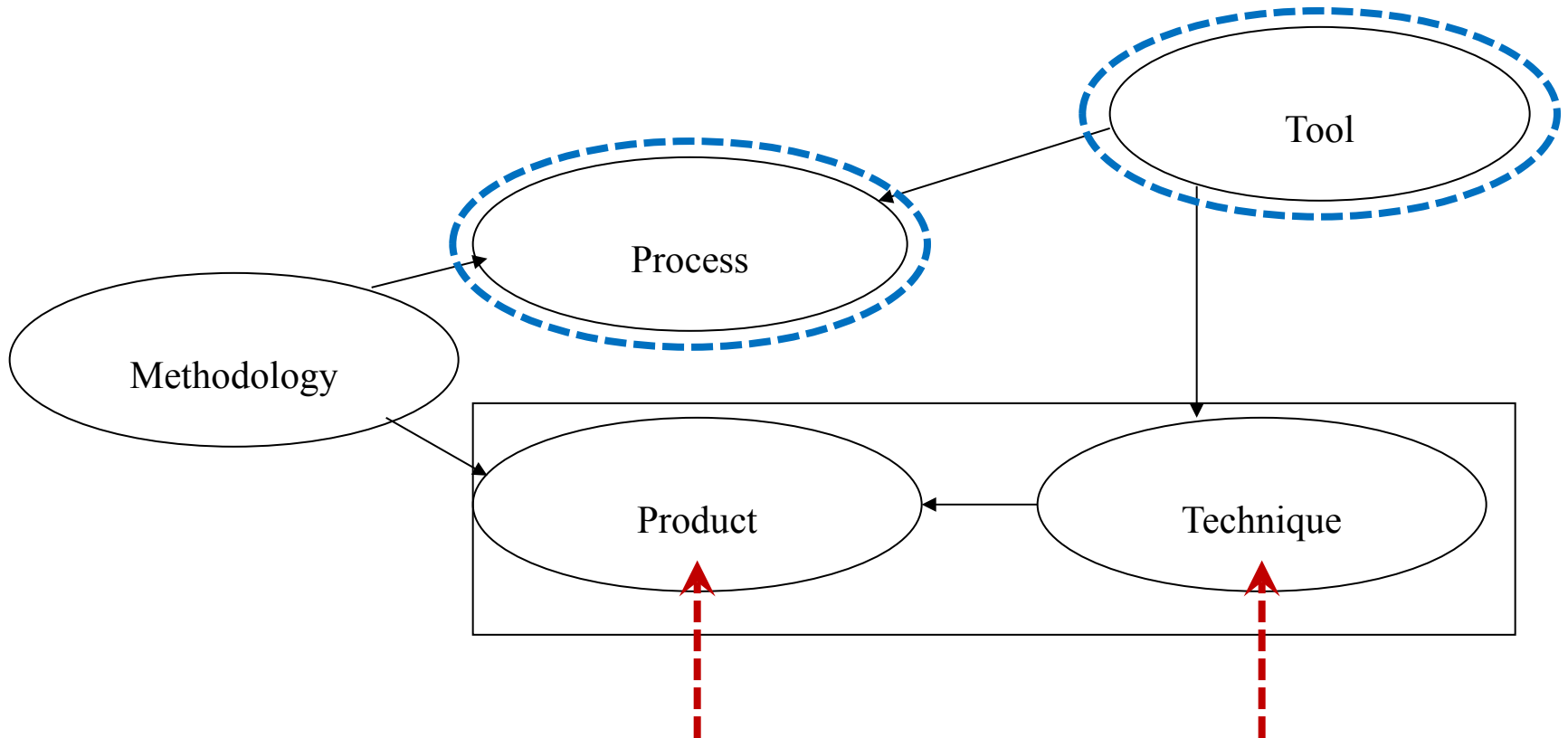


Development Methodologies

- “An IS methodology is a methodical (*systematic*) approach to IS planning, analysis, design, construction and evolution.” (Olle, 1988)
- More than 1000 exist...
 - BON, Booch, BOOM, Catalysis, CBD/e, Coad/Yourdon, COMMA
 - CRC, Convergent Engineering, Demeter, DOORS, DOOS
 - EPA, EROOS, Fusion, Goofee, HOOD, IDEA, ION, KISS
 - MERODE, MOSES, MWOOD, Object COMX, Objecteering
 - Objectory, OEP, Octopus, OMT, OOAD/OOIE, OOA/RD, OOBE
 - OOCL, OOHDM, OOram, OOSC, OOSD, OOSE, OOSP
 - Open, OSA, PAUD, ROAD, ROPES, RUP, Scrum, Skill-Driven Design
 - SDL, Shlaer & Mellor, Softstar, SOMA, SOMT, Syntropy, XP



Taxonomy of Methodologies





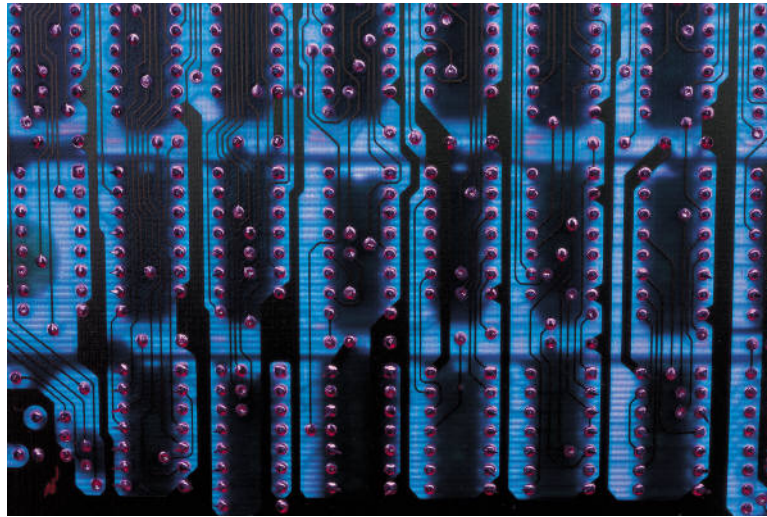
Issues with Methodologies

- Adoption:
“Many organizations claim that they do not use any systems development methods.” (Huisman & Iivari, 2002)
- Vagueness:
 - “Low coupling” is still vague, “Information hiding” was formulated in 1972 (Parnas), but still needs refining
 - “We haven’t found the fundamental laws in software like in other engineering disciplines” (Kruchten, 2005)
- Limited, unsystematic application:
 - Technical difficulties
 - Project management difficulties

Use Engineering Fundamentals

- Stability in system dynamics:
 - Confining the impact of adding or modifying modules is basically a systems stability issue
- Entropy in statistical thermodynamics:
 - Narrowing down the microscopic cause of a macroscopic error is basically an entropy issue
- Hierarchical modular architectures:
 - Having an integrated zoomable view on billions of constituent parts is basically a modularity issue

Use Engineering Fundamentals



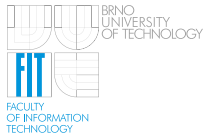
Other disciplines have mastered the *unambiguous hierarchical assembly structure of large amounts of fine-grained static modules ...*

Quo Vadis Software Engineering?

Expectations and Reality

Radek Kočí

Brno University of Technology, Faculty of Information Technology
Božetěchova 2, 612 66 Brno, CZ
koci@fit.vutbr.cz



What does customer expect?

- releases are valid and satisfy *all* needs
- the terms are kept (deadlines, budget, ...)
- *the customer does not care about development process*

What does developer expect?

- releases are valid and satisfy *reasonable (minimal)* set of needs
- the terms are kept (deadlines, budget, ...)
- *the developer should care about development process*

Questions

- the customer needs not to care about processes – really?
- the developer needs not to care about real needs – really?
- are there common techniques used in methods that keep to the terms?

The customer/developer needs not to care – really does not?

- it is not truth
- nowadays, the problem domains and the development processes are very complex and we need to have a knowledge
- the key activity – *we need to care how to get real (valid) requirements*

Are there common techniques for development methods?

- iterative and incremental development processes
- modeling
- automated model transformations, code generation
- model continuity – executable models that can be deployed as a part of target system

What do students (a lots of them) expect?

- software engineering = *many words about nothing*
- the best approach is *agile approach*, whereas agile means chaotic approach having no rules
- it is good to know about UML

What/How should we teach?

- timeless principles
 - how to separate timeless principles from outdated ones?
- cohesion of teaching and software engineering research and practice
 - to get the best practices
 - participation in research, practice – is it realizable?
- helping students learn how to learn
 - participation in research, practice – is it realizable?
- students have to be active in real problems – how to do it?