Internet of Things - Engineered
What’s feasible?

Carl Worms
Softwareentwicklung in der Industriellen Praxis

Contents

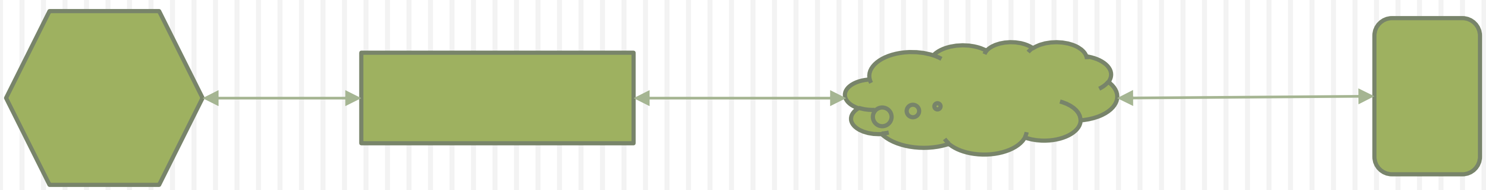
- WhoamI
- Internet of Things – What's new
- Software Engineering Nowadays
- What do Other Engineering Disciplines?
- Software Engineering Advanced
- What's needed?
- Who dares?

Who am I

- 1975- : Computer Science in Karlsruhe, Germany
- 1978- : Lived from programming for 20 years
- 1991- : Software Quality/Testing
- 1993: Walter Masing Awardee (DGQ)
- 1999: IT Architect/SWE Process Architect at a major Swiss bank for 15 years
- 2007- : PC member of IEEE conferences, Keynotes, Papers
- Member of GI, DGQ, IEEE

Internet of Things - What's New?

Overall Architecture [1]



Devices
- Actuators
- Sensors
- Tags

Gateways

Cloud
- Device registry
- Sensor data storage
- Domain algorithms and analytics

Apps and visualizations

Internet of Things - What's New?

- What makes IoT development different [1]:
  - IoT devices are just a tiny part of a larger system
  - IoT systems never sleep or shut down in entirety
  - IoT systems are more like cattle than pets
  - IoT devices are often embedded in surroundings and such physically invisible and unreachable
  - IoT systems are highly heterogeneous
  - IoT systems tend to have weak and unreliable connections
  - IoT system topologies can be highly dynamic and ephemeral

Internet of Things - What‘s New?

- Challenges for software development [1]:
  - Multidevice programming
  - The reactive, always-on nature of the system
  - Heterogeneity and diversity
  - The distributed, highly dynamic and migratory nature of software
  - The general need to write software in a fault tolerant and defensive manner

Internet of Things - What‘s NOT New?

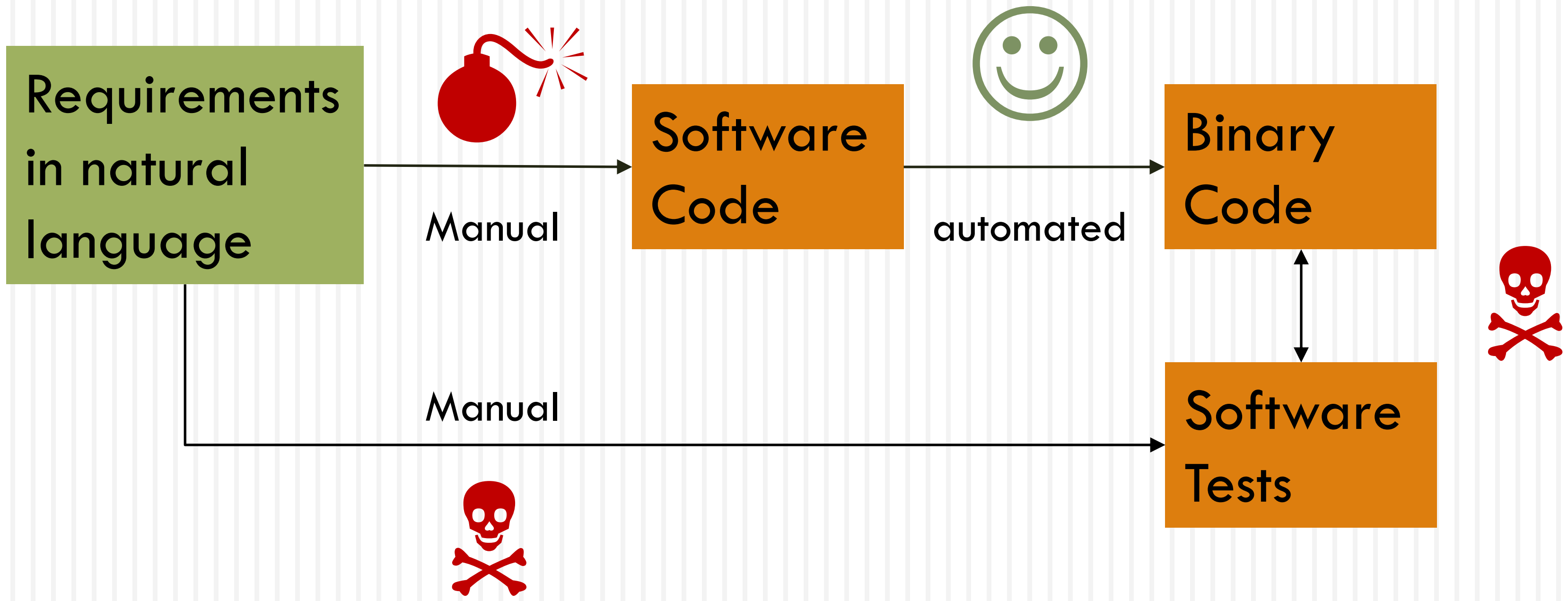
- Eight false assumptions of programmers when writing software for distributed systems [2]:
  - The network is reliable
  - Latency is zero
  - Bandwidth is infinitive
  - The network is secure
  - Topology doesn’t change
  - There is one administrator
  - Transport cost is zero

Internet of Things - What‘s New?

- Various implications [2]:
  - Improper balance between application logic and error code
  - Underestimated costs of building and maintaining software
  - Inadequate languages and tools (e.g. JavaScript), which don’t address programming-in-the-large, support orchestration of large systems or flexible migration of code
  - Security risks, e.g. thousands of IoT devices still having their standard security settings incl. the default admin password
- Need for appropriate software engineering technologies, methodologies, abstractions, etc.

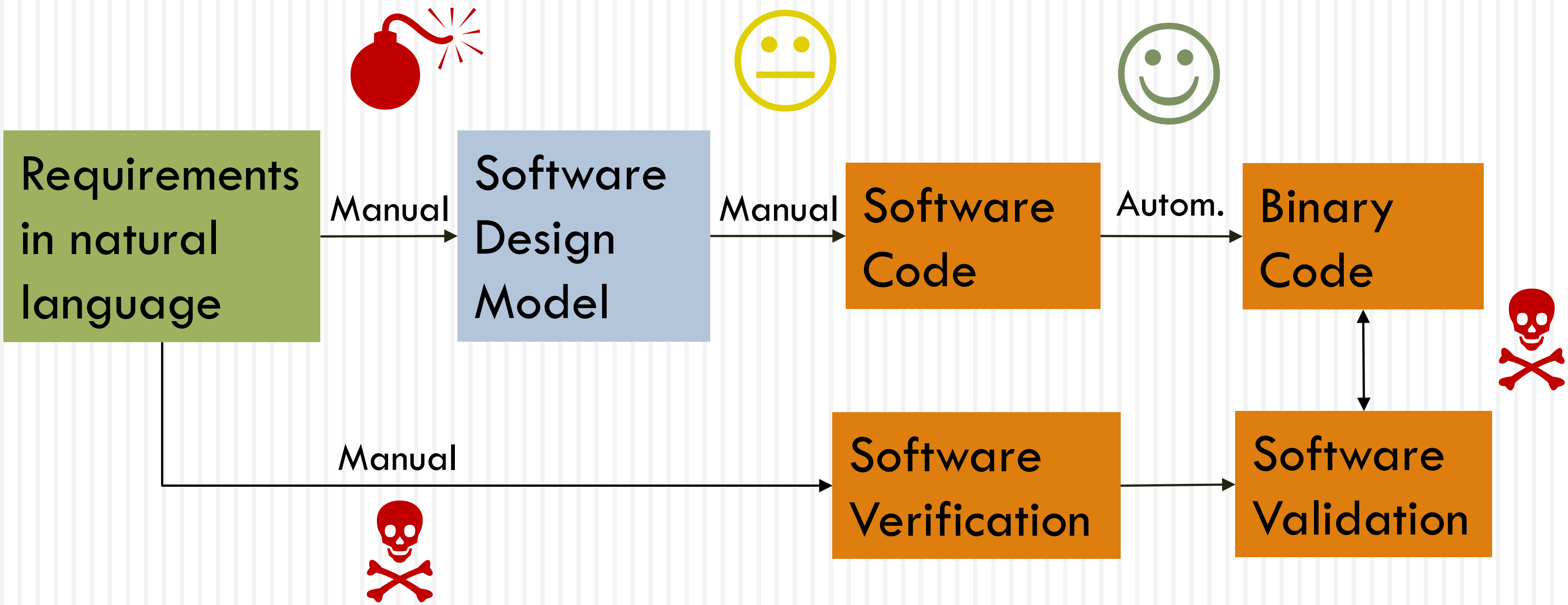
Software Engineering - Nowadays

Very popular:



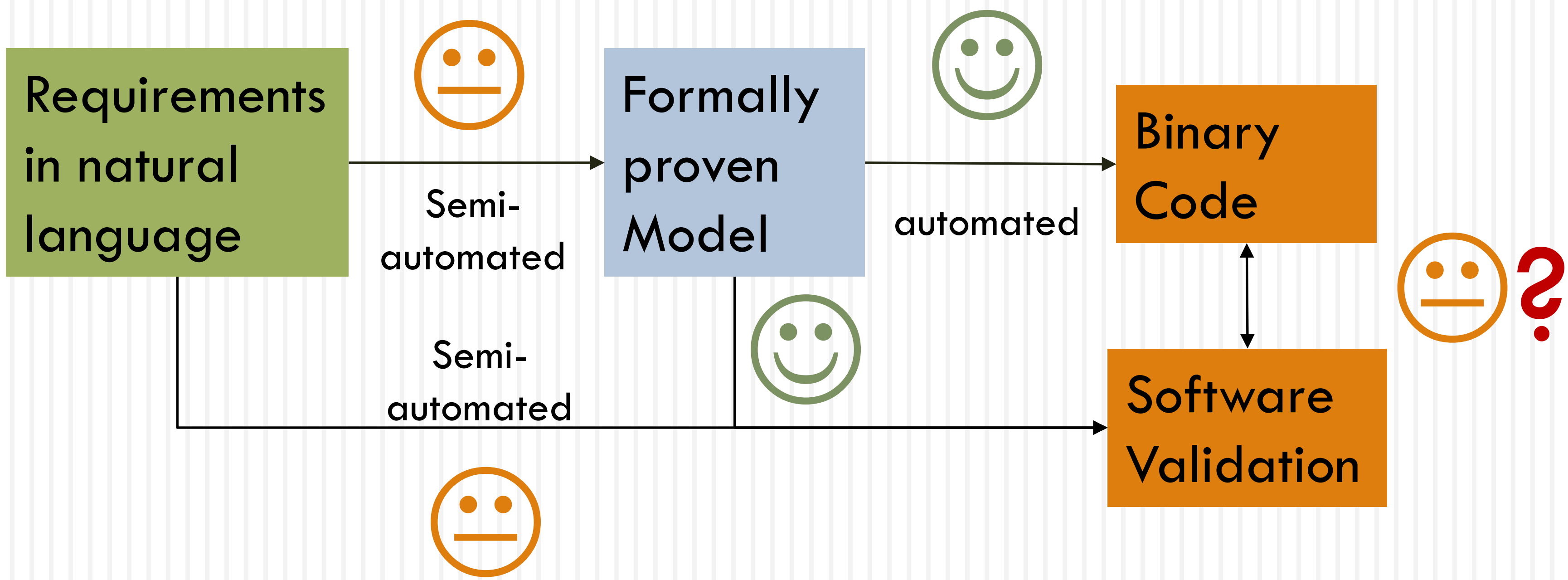
Software Engineering - Nowadays

«Professional»:



Software Engineering - Nowadays

Very rare:



Software Engineering - Old Facts

- Software Defect Reduction Top 10 List [12]:
  1) *Finding and fixing a software problem after delivery is often 100 times more expensive than finding and fixing it during the requirements and design phase;* for small, noncritical systems it is more like 5:1
  2) *Software projects spend about 40 to 50 % of their effort on avoidable rework*
  3) *About 80% of avoidable rework comes from 20% of the defects* (lower for smaller, higher for very large ones)
  4) *About 80%* (median) *of the defects come from 20% of the modules, and about half the modules are defect free*
  5) *About 90% of the downtime comes from, at most, 10% of the defects*

Software Engineering - Old Facts

- Software Defect Reduction Top 10 List [12]:

6) *Peer reviews catch 60% of the defects*
7) *Perspective-based reviews catch 35% more reviews than nondirected reviews*
8) *Disciplined personal practices can reduce defect introduction rates by up to 75%*
9) *All other things being equal, it costs 50% more per source instruction to develop high-dependability software products than to develop low-dependability software products. However, the investment is more than worth ist if the project involves significant maintenance and operations cost.* Low-dependability software costs about 50% per instruction more to maintain than to develop, whereas high-dependable software costs 15% less. For a typical life-cycle cost distribution of 30% development and 70% maintenance, both software types become about the same in cost […]
10) *About 40-50% of user programs contain nontrivial defects.* Between 21 and 26% of operational spreadsheets contain defects.

Software Engineering - Nowadays

- Other observations after 60 years of SWE:
  - Error-prone number entry in e.g. medical devices [3]
  - Still 'bare-metal programming' (without IDE) for embedded or safety-related software [4]
  - Quality of Service (QoS) of distributed systems only partially matches with the latest software quality standard ISO/IEC 25010 [5][6]
  - A new hot spot of QoS is energy consumption [7][8][9]
  - Internet App research with concerning results [10]

Software Engineering - Nowadays

Real Engineering practice

- Well-codified knowledge, preferentially scientifically-founded, shapes design decisions
- Reference materials make knowledge and experience available
- Analysis of a design predicts properties of ist implementation

SW Engineering status

😐 We have some guidance for design decisions, but not nearly enough nor systematic enough

☹ Reference materials and documen-tation are widely neglected. We have scientific papers, […] and searchable APIs for specific systems – but well curated reference are sorely lacking

😐 We have a rich set of analysis technics, but <u>most focus on the code</u> rather than the design. We have rich simulations systems in certain areas. But we still lack […] exploring design alternatives before implementation [11]

Software Engineering - Nowadays

What are your pros and cons regarding present software engineering?

What's missing?

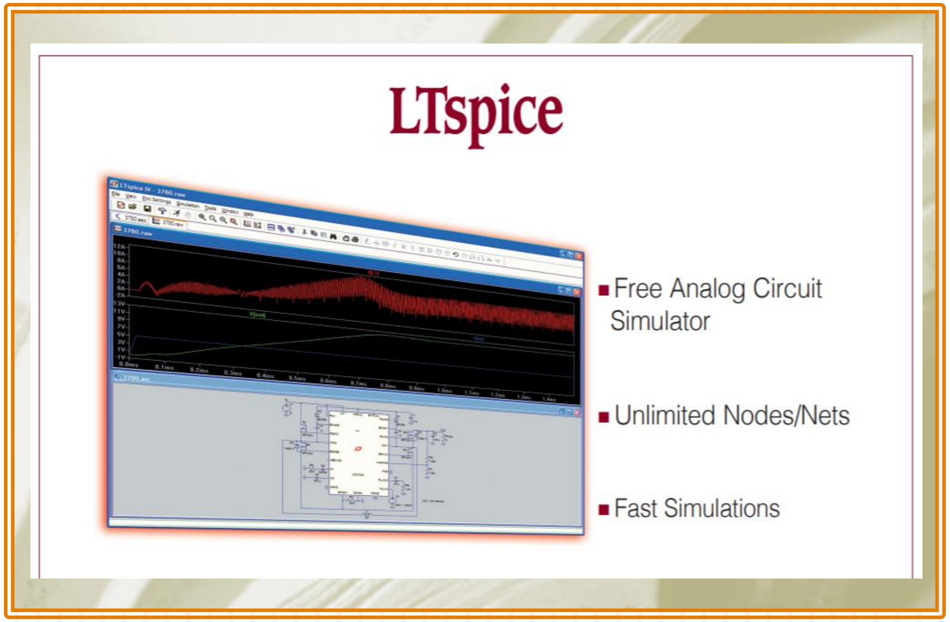
What Do Other Disciplines?

- Mechatronics (easy):
  - Use e.g. *Fritzing*
  - Use domain specific part collections (via standardized interfaces)
  - Use domain specific simulation
  - Build the system really

Fritzing Intro

What Do Other Disciplines?

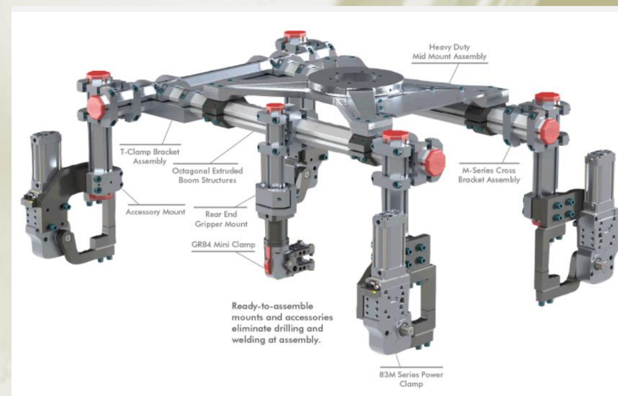
- Electronics (for Pro's):
  - Use e.g. *LTSPICE* (since 20 years)
  - Use domain specific part collections (via standardized interfaces)
  - Use domain specific simulation
  - Build the system really



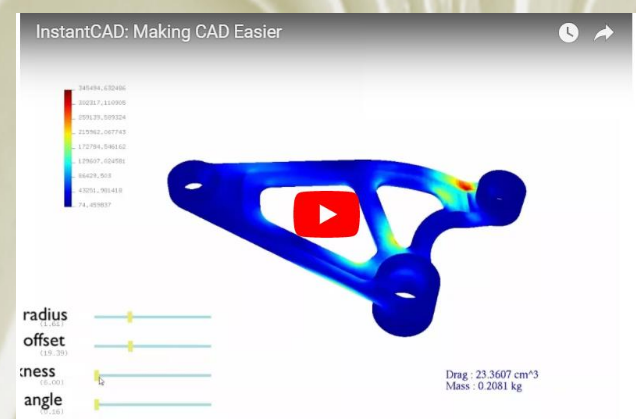
[LTSPICE Overview]

What Do Other Disciplines?

- Mechanics:
  - Use Computer Aided Design (CAD)
  - Use domain specific part collections (via standardized interfaces)
  - Use domain specific simulation (e.g. finite elements)
  - Build the system really



Destaco BodyBuilder



MIT InstantCAD

What Do Other Disciplines?

- Civil Engineering:
  - Define domain specific targets
  - Use Computer Aided Design (CAD)
  - Use domain specific simulation
  - Connect with other IT systems
  - Build the system

Präsentation Hochschule Luzern

What Do Other Disciplines?

- Summary
  - Design: CA* tools and part collections including all relevant physical parameters for the domain, based on formal methods and empirical natural sciences
  - Process: design and verify/validate with domain-specific software, than build
  - People: only accept formal education and certificates
  - Education: teach math adapted for the discipline
  - Research: focus on new physics/materials/simulations
  - Regulators: improve and develop standards/rules

What Do Other Disciplines NOT?

- Summary
  - Process: do what you like
  - People: accept experience as replacement for formal education and certificates
  - Education: teach math not applied for their discipline
  - Research:
    - Mix of the core discipline and business analysis/operations
    - E.g. observe communication between designers to find out properties of the parts they work on

IoT - Software vs. Other Engineering

- Personal conclusion:
  - SWE maturity after 60 years is probably similar than mechanics and civil engineering after 60 years – who remembers broken gothic churches or bridges from many years ago or exploding steam engines (sometimes explode chemical plants even in Europe and the US …)
  - Internet of Things bears the clash of quite different maturities between SWE vs. Mechatronics – if regulations and quality expectations don't decrease too much, this will force SWE to higher maturity

Software Engineering - Advanced

What further progress could SWE make?

Your ideas?

Software Engineering - Advanced

- Topics [13]:
  - Verification of physical systems as they work in the real world
  - Formal methods will be a key enabling technology
  - SWE … has become more about the composition of existing functionality while adding some innovative functions …
  - … new strategies to blend traditional testing, new advances in formal methods, modeling and simulation and automated testing, and continued data collection after fielding.

Software Engineering - Advanced

- Composition of existing functionality
  - Zhu, Bayley [16]: Composition of design patterns
  - Jatoth et al. [17]: Literature Review on QoS-Aware Web Service Composition
  - Andreou, Papatheocharous [25]: Automated matching of component requirements
- New advances in formal methods:
  - Abrial [18][19]: Event-B method and toolset, industrial applied in
    - Railway engineering [20]
    - Real Time Operating System Memory Manager [21] (an excellent example of the application of Event-B)
  - Morales, Capel [22]: Model checking for critical systems

Software Engineering - Advanced

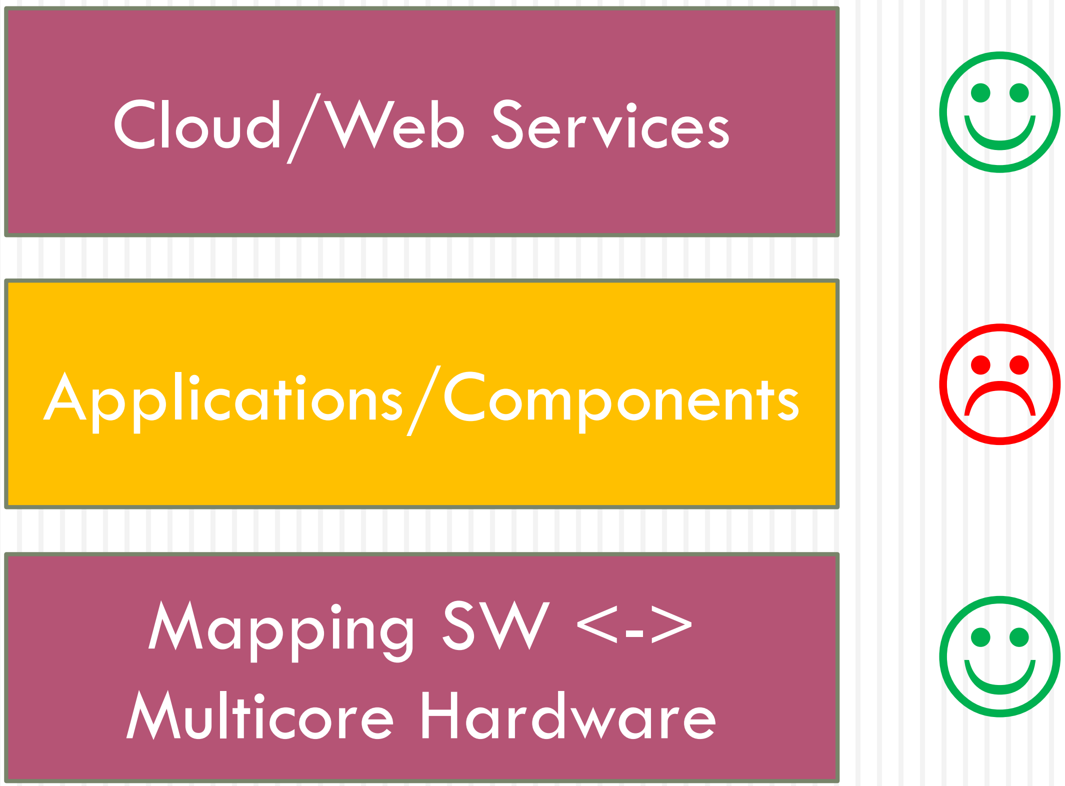
- Modeling
  - ThingML approach for IoT [14]
  - IoT Reference Architectures [15] and comparison
- Code generation
  - On-the fly for scientific computing [23]
  - Safety-critical avionics software [24]
- Simulation
  - Comparison of performance prediction methods [25]
- Etc., etc.

Software Engineering - Advanced

- Missing
  - Domain-specific standard sets of a software components runtime parameters
  - E.g.:
    - Min/mid/max response time
    - Consumption of CPU/storage/network on a reference platform/in a reference network
    - Correctness proven yes/no
    - …

Software Engineering - Advanced

- Interesting: focus of QoS practice and research



What‘s needed?

- Education:
  - Math lectures (logic, set theory, statistics) adapted to software engineer‘s needs
  - Tutorials/exercises in formal methods and present tool sets
- Research:
  - Improvement of formal methods and tools for large distributet systems
  - Refocus on Software Empirics vs. the Software Engineer
- Industry: the «Innovative Formal Guerilla»

Who dares to …?

… develop formal correct Linux drivers?

… develop the first formal proven App?

… develop a formal correct Linux FC 1.0?

… develop a better RODIN for students?

… found a commercial company to produce formal proven only systems and software?

A Last Word

Thank you

References

[1] A. Taivalsaari, T. Mikkonen, " A Roadmap to the Programmable World. Software Challenges in the IoT Era"; *IEEE Software*, Januar/February 2017

[2] A. Rotem-Gal-Oz, "Fallacies of Distributed Computing Explained"; www.rgoarchitects.com/Files/fallacies.pdf

[3] H. Thimbleby, "Safer User Interfaces: A Case Study in Improving Number Entry"; *IEEE Trans. on Softw. Eng.*, Vol. 41, No. 7, July 2015

[4] G. H. Holzmann, "Tiny Tools"; *IEEE Software*, January/February 2016

[5] J. Kiruthika, S. Khaddaj, "Software Quality Issues and Challenges of Internet of Things"; *2015 14th International Symposium on Distributed Computing and Application for Business Engineering and Science*, pp. 176-179, 2015

[6] T. Bianchi, D. S. Santos, and K. R. Felizardo, "Quality Attributes of Systems-of-Systems: A Systematic Literature Review"; *2015 IEEE/ACM 3rd International Workshop on Software Engineering for Systems-of-Systems (SESoS)*, pp. 23-30, 2015

[7] K.-Y. Chen, J. M. Chang, and T.-W. Hou, "An Energy-Efficient Java Virtual Machine"; *IEEE Trans. on Cloud Computing*, vol. 5, pp. 263-275, April-June 2017

[8] S. Wang, A. Zhou, C.-H.Hsu, X. Xiao, and F. Yang, "Provision of Data-Intensive Services Through Energy- and QoS-Aqare Virtual Machine Placement in National Cloud Data Centers"; *IEEE Trans. on Emerging Topics in Comp.*, vol. 4, no. 2, June 2016

References

[9] M. Wan, Y. Jin, D. Li., and W. G. Halfond, "Detecting Display Energy Hotspots in Android Apps"; in *Proc. IEEE 8th Int. Conf. Softw. Testing, Verification and Validation,* pp. 1-10, 2015

[10] W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harmann, "A Survey of App Store Analysis for Software Engineering"; *IEEE Trans. on Softw. Engineering*, vol. 43, no. 9, September 2017

[11] M. Shaw, “Progress Toward an Engineering Discipline of Software (Keynote)“; *2016 IEEE/ACM 38th Int. Conf. on Softw. Eng. Compagnion*, p. 3

[12] Victor R. Basili, Barry Boehm, "Software Defect Reduction Top 10 List", *Computer*, vol. 34, pp. 135-137, January 2001

[13] A. Moore, T.O’Reilly, P. D. Nielsen, and K. Fall, “Four Thought Leaders on Where the Industry is Headed“; *IEEE Software*, pp. 36-39, January/February 2016

[14] B. Morin, N. Harrand, and F. Fleurey, “Model-Based Software-Engineering to Tame the IoT Jungle“; *IEEE Software*, pp. 30-36, January/February 2017

[15] M Weyrich, C. Ebert, “Reference Architectures for the Internet of Things“; *IEEE Software*, pp. 112-116, January/February 2016

[16] H. Zhu, I. Bayley, “On the Composability of Design Patterns“; *IEEE Trans. On Softw. Eng.*, vol. 41, no. 11, November 2015

[17] C. Jatoth, G. R. Gangadharan, and R. Buyya, “Computational Intelligence Based QoS-Aware Web Service Composition: A Systematic Literature Review“; *IEEE Trans. On Services Comp.*, vol. 10, no. 3, May/June 2017

References

[18] J.-R. Abrial, “Faultless Systems: Yes, we can!“, *Computer*, pp. 30-36, September 2009

[19] J.-R. Abrial, "Formal Methods in Industry: Achievements, Problems, Future", *Software Engineering, International Conference on*, pp. 761-768, 2006

[20] T. Fischer, “Rodin in the Field of Railway System Engineering“; *6th Rodin User and Developer Workshop 2016*, http://wiki.event-b.org/index.php/Rodin_Workshop_2016

[21] W. Su, J.-R. Abrial, G. Pu, and B. Fang, “Formal Develoment of a Real-Time Operating System Memory Manager“; *2015 20th Int. Conf. On Eng. Of Compl. Comp. Syst.*, 2015

[22] L. E. Mendoza Morales, M. I. Capel, “Checking Critical Software Systems: A Formal Proposal“; *2016 10th Int. Conf. On the Quality of Inf. and Comm. Techn.*, 2016

[23] A. Heinecke, G. Henry, M. Hutchinson, and H. Pabst, “LIBXSMM: Accelerating Small Matrix Multiplications by Runtime Code Generation“; *SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2016

[24] A. Wölfl, N. Siegmund, S. Apel, H. Kosch, J. Krautlager, and G. Weber-Urbina, “Generating Qualifiable Avionics Software: An Experience Report“; *2015 30th Int. Conf. on Autom. Softw. Eng.*, 2015

[25] A. S. Andreou, E. Papatheocharous, “Automatic Matching of Software Component Requirements Using Semi-Formal Specifications and a CBSE Ontology“; *2015 International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*