Software Quality – Promise or Threat?

Carl Worms
Softwareentwicklung in der Industriellen Praxis

Contents

- WhoamI
- Software Quality – State of the Art
- Software Quality – the Main Problem
- 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 16 years
- 2007- : PC member of IEEE conferences, keynotes, papers, lectures
- Member of GI, DGQ, IEEE

Software Quality – State of the Art

Definition [1]:

The quality of a system is the degree to which the system satisfies the stated and implied needs of its various stakeholders, and thus provides value.

Software Quality – State of the Art

History [2]:

Germination stage (1970-1990):
- concept of software quality, factors, evaluation

Exploration stage (1990 – 2001:
- SW product evaluation, quality metrics
- ISO/IEC 14598, ISO/IEC 9126

Mature stage (since 2001): SQuaRE

Software Quality – State of the Art

SQuaRE: **S**ystems and software **Qua**lity **R**equirements and **E**valuation, structure of standards

Quality management	ISO/IEC 2500n
Quality model	ISO/IEC 2501n
Quality measurement	ISO/IEC 2502n
Quality requirement	ISO/IEC 2503n
Quality evaluation	ISO/IEC 2504n
Extension	ISO/IEC 25050 - 25099

Software Quality – State of the Art

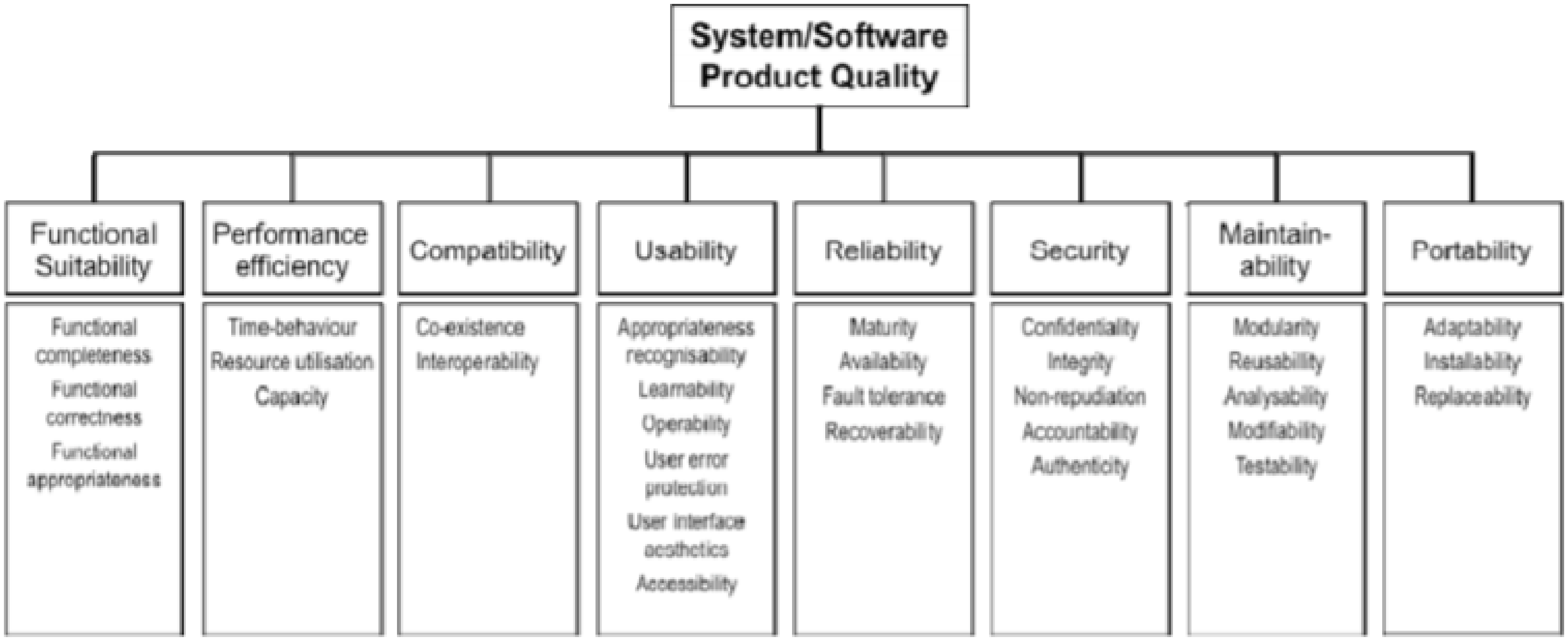
Quality model, general structure

Quality =

- Sum of characteristics =
  - Sum of subcharacteristics =
    - Sum of quality properties =
      - Sum of quality measure elements

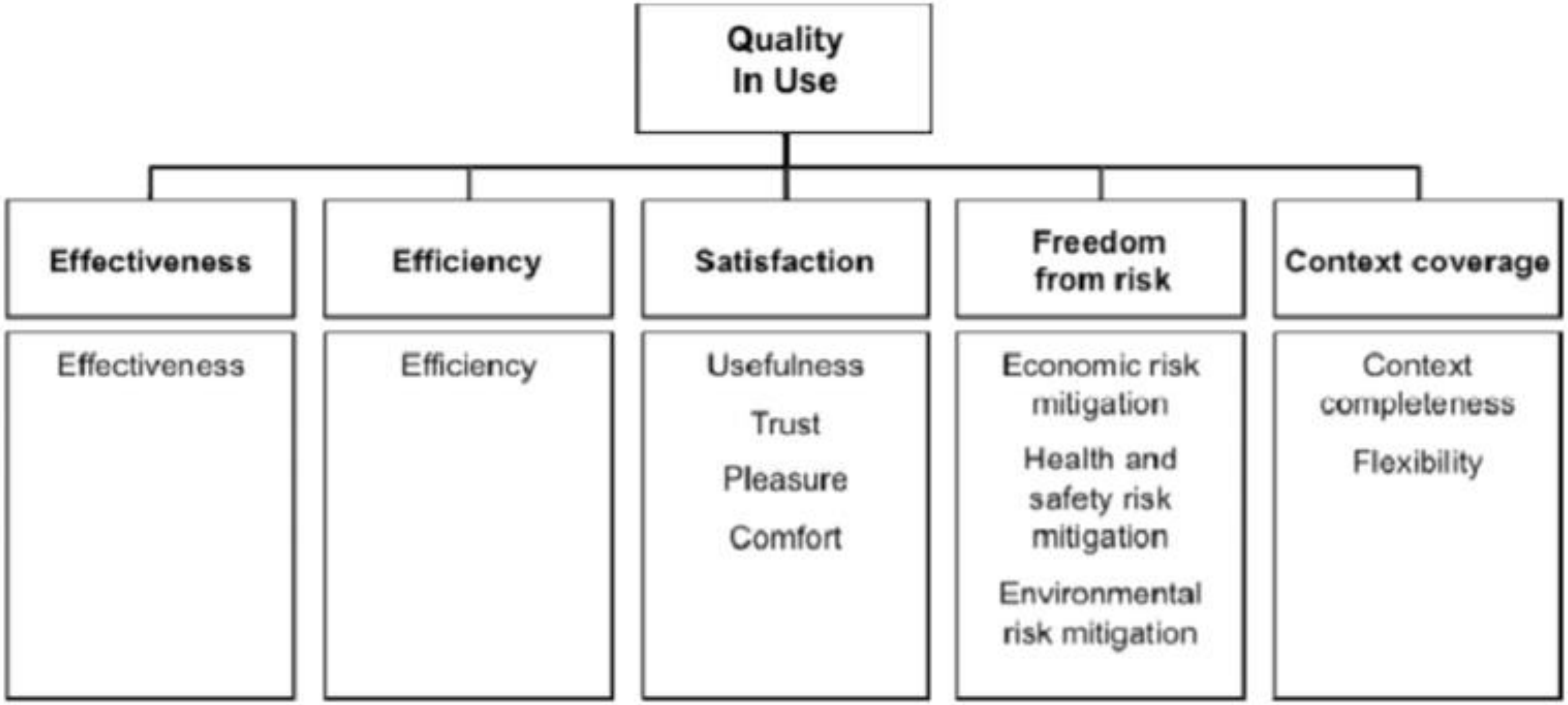
Software Quality – State of the Art

Systems and software product quality model [1]



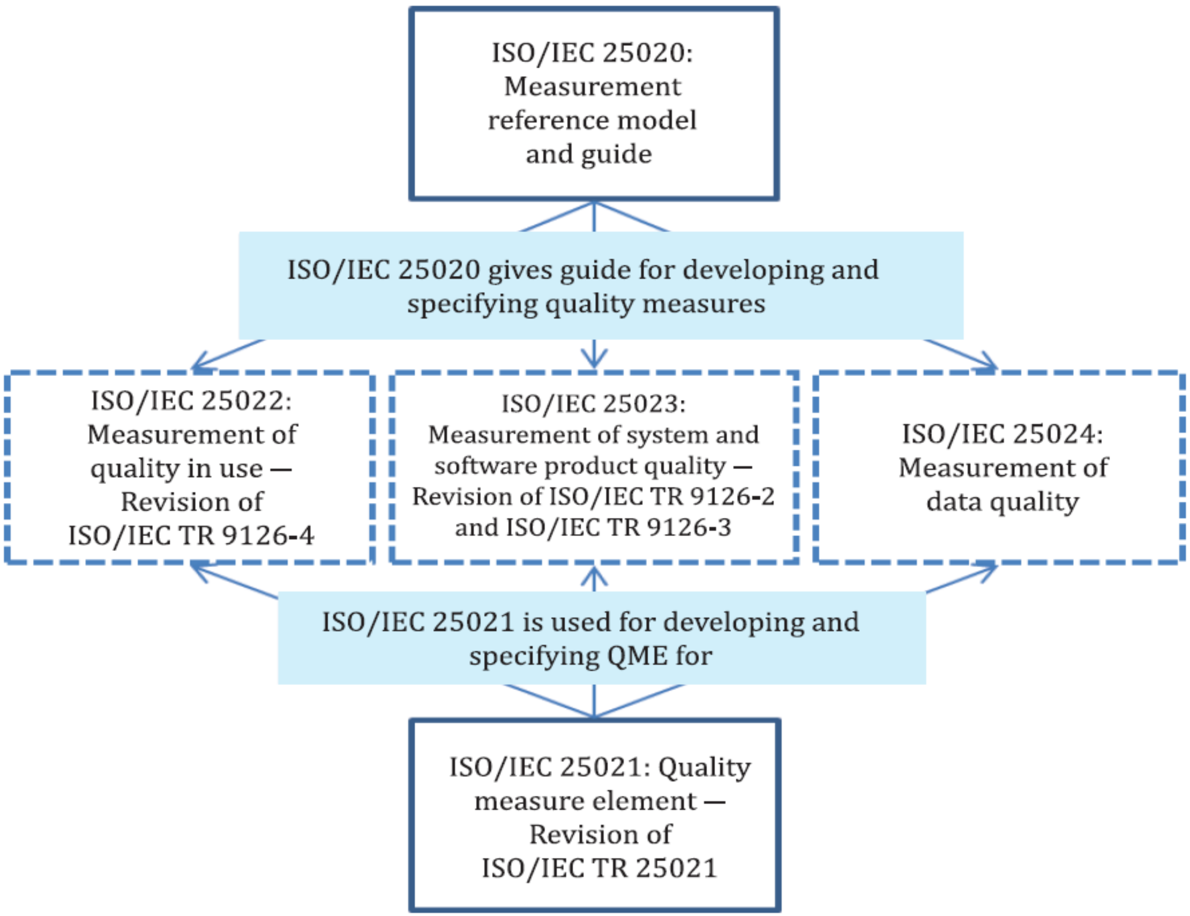
Software Quality – State of the Art

Quality in use model [1]



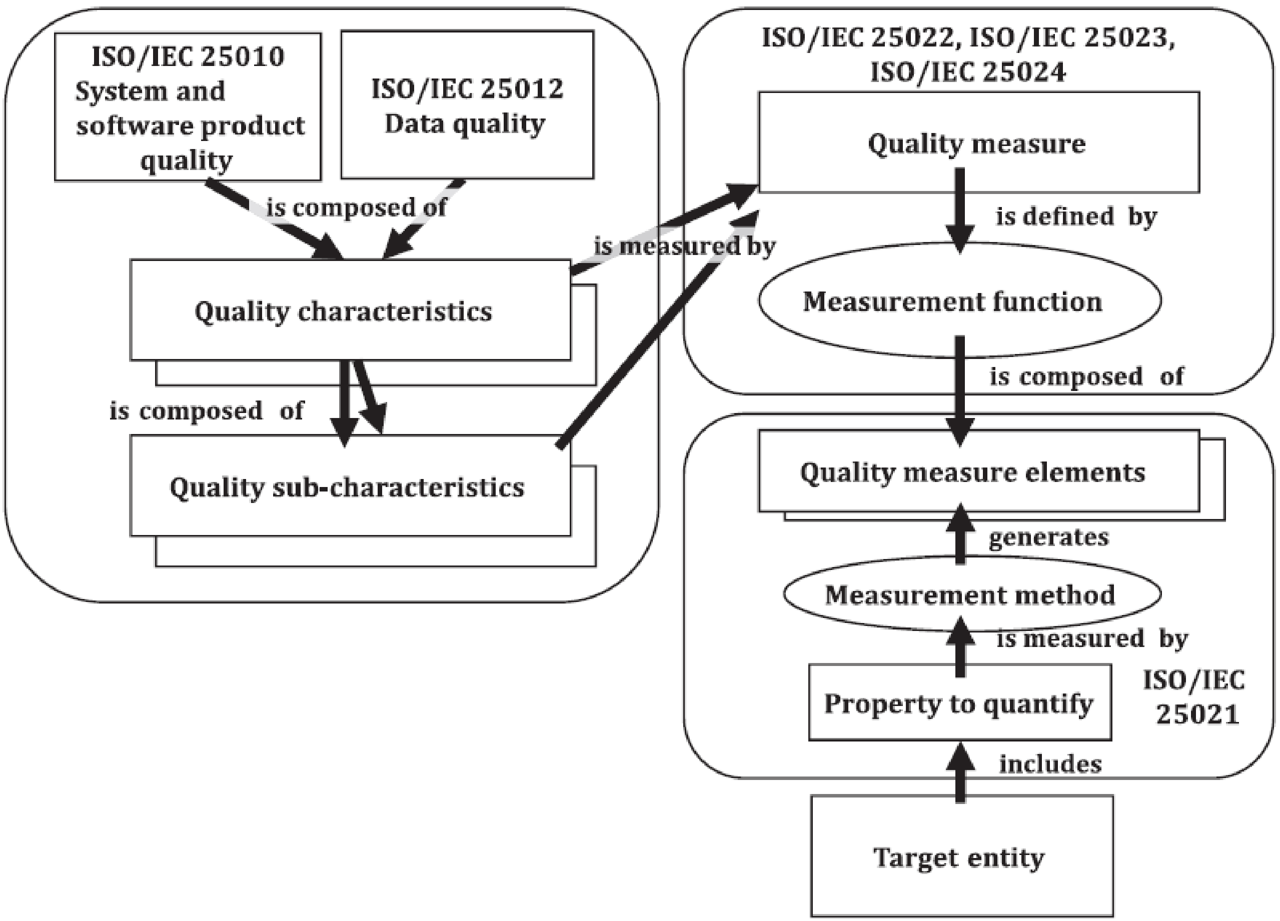
Software Quality – State of the Art

Quality Measurement standards overview [3]



Software Quality – State of the Art

Relationship among quality model and measure [3]



Software Quality – State of the Art

Present reasearch on SW quality (examples):

- Challenges of overall quality evaluation [4]
- Quality Trade-offs in Embedded Systems [16]

- Realistic failure models of SW components [5]
- Data quality models for web portals [6]

- Empirical studies on quality prediction [7]
- Simulation of software quality [8][9]

Software Quality – State of the Art

Present reasearch on factors impacting SW quality:

- Requirements Traceability Completeness [10]
- Architectural Technical Debt [11]
- Object-Oriented Code Refactoring [12]
- Classification of poor data quality [13]

- Quality assurance for big data applications [14]
- Testing of Concurrent Software Systems [17]

- Organizational parameters as quality predictor [15]
- SW quality and agile methods -> see other lectures

Software Quality – State of the Art

Present-day issues:

- Error-prone number entry in e.g. medical devices [18]
- Still 'bare-metal programming' (without IDE) for embedded or safety-related software [19]

- Quality of Service (QoS) of distributed systems only partially matches with the latest software quality standards ISO/IEC 25010 [20][21]
- A new hot spot of QoS is energy consumption [22][23][24]
- Internet App research with concerning results [25]

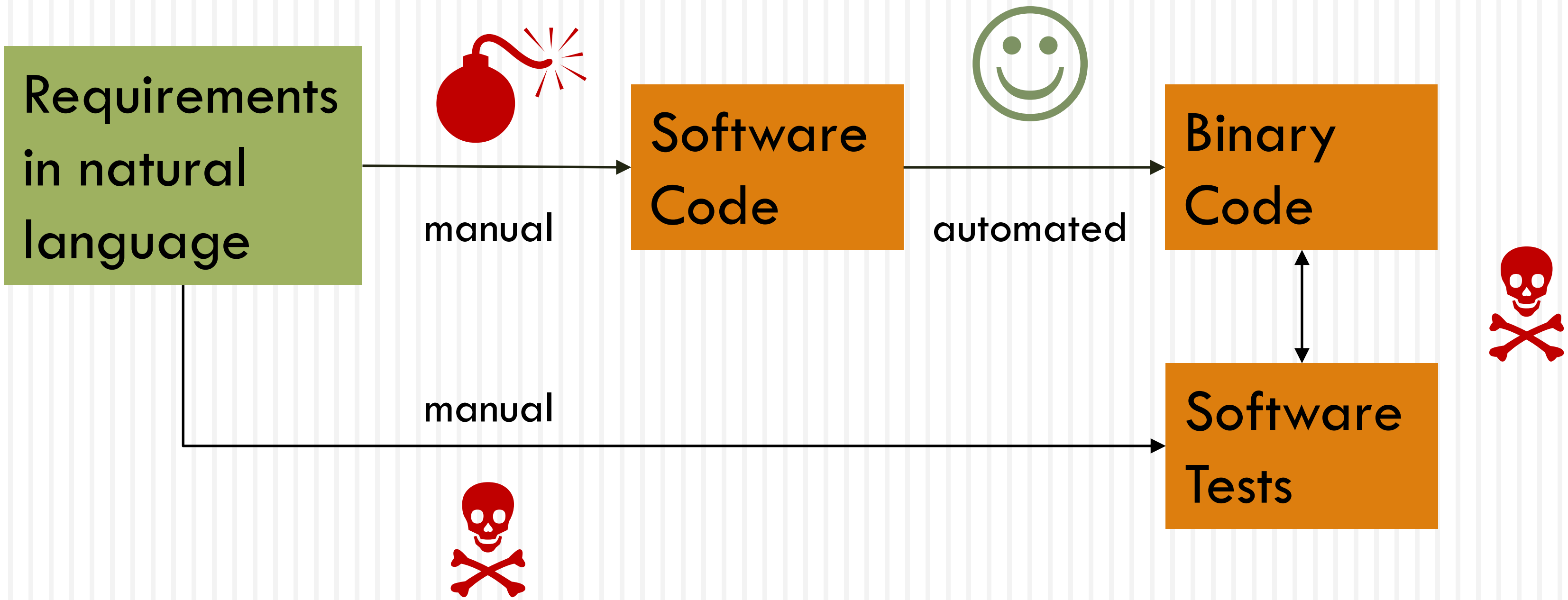
Software Quality - theProblem

What are your pros and cons regarding present software quality?

What's missing?

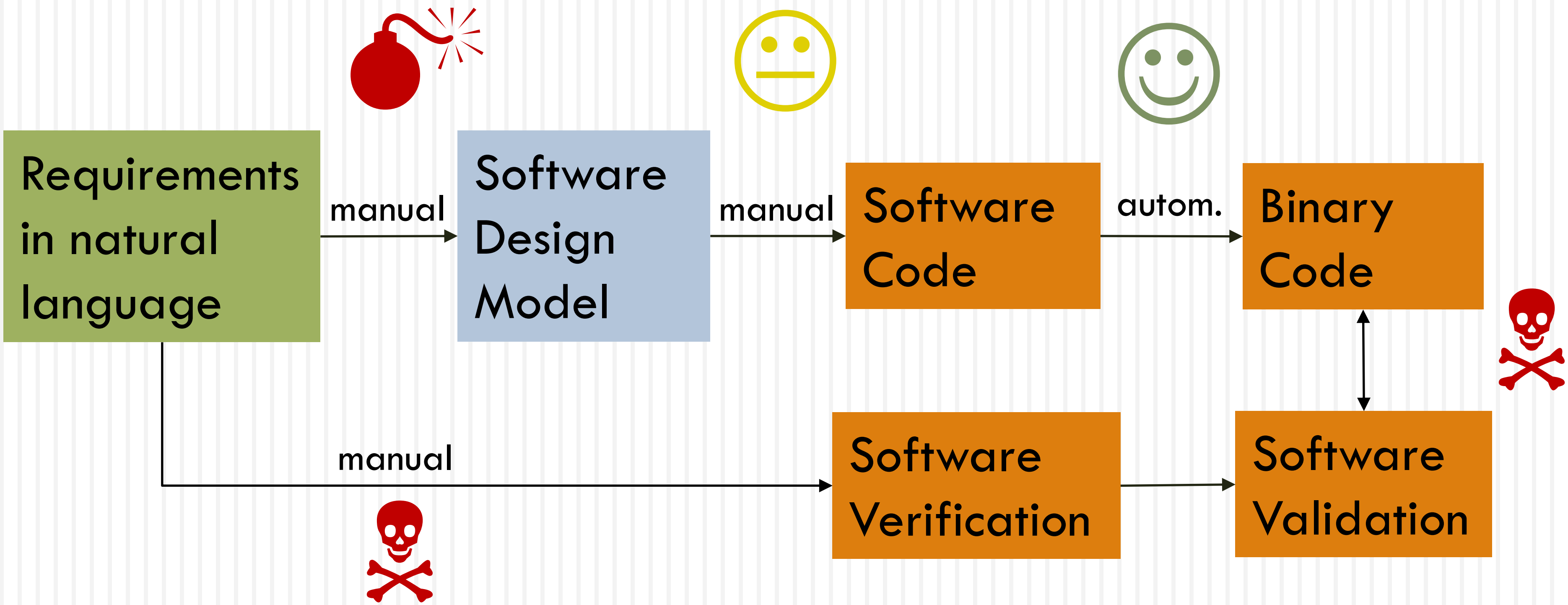
Software Quality – the Problem

Very popular:



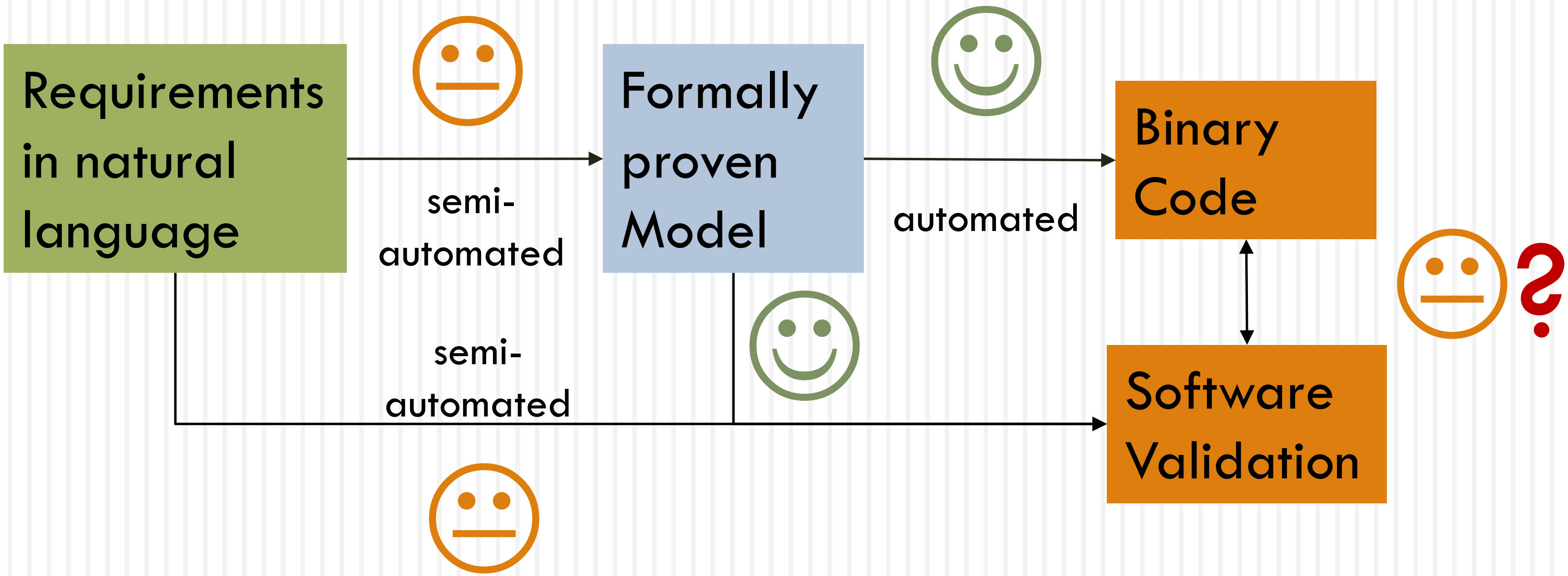
Software Quality – the Problem

«Professional»:



Software Quality – the Problem

Very rare:



Software Quality - Old Facts

- Software Defect Reduction Top 10 List [27]:
  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 Quality - Old Facts

- Software Defect Reduction Top 10 List [27]:
  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

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 its 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 most focus on the code rather than the design. We have rich simulations systems in certain areas. But we still lack […] exploring design alternatives before implementation [26]

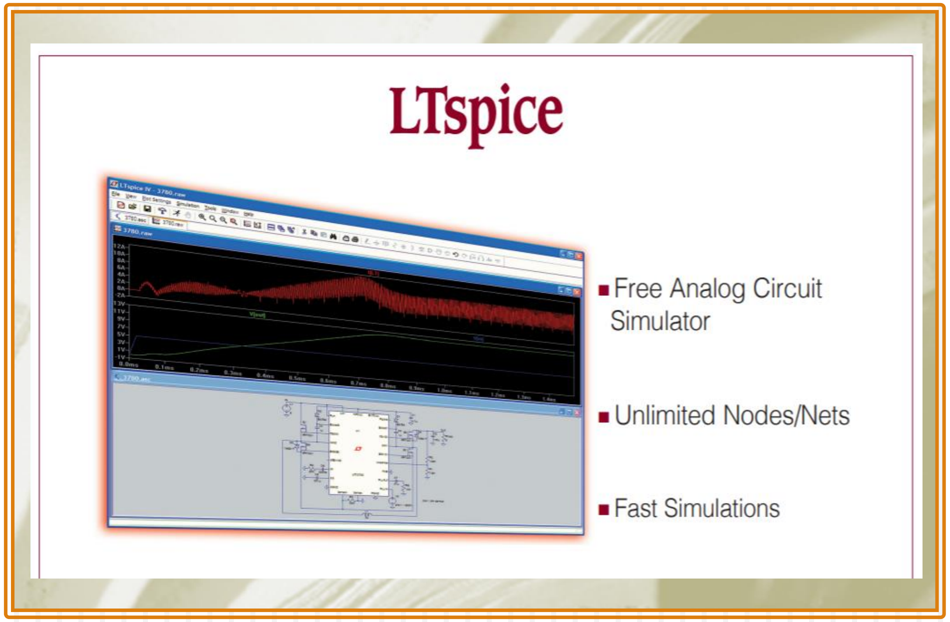
What Do Other Disciplines?

- 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?

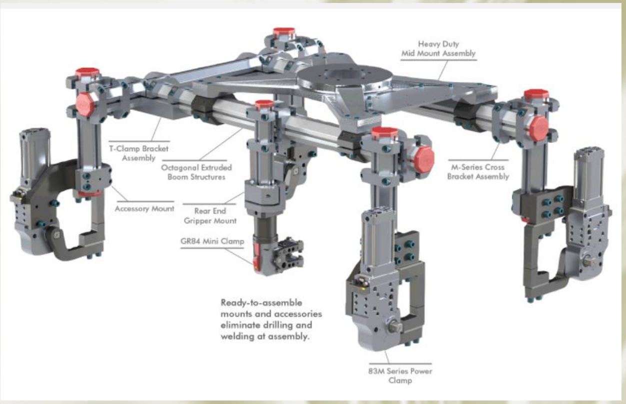
- Electronics (for Pro's):
  - Use e.g. *LTSPICE* (since 20 years)
  - Use domain specific part collections (via standardized interfaces)
  - Use domain specific simulation
  - Build the system really



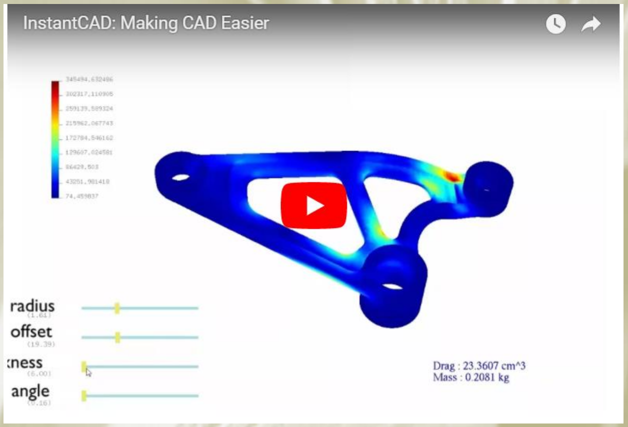
LTSPICE Overview

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 <u>including all relevant physical parameters</u> 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, tweak standards
  - People: accept practical 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 or psychology
    - E.g. observe communication between designers to find out properties of the parts they work on

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 still explode chemical plants even in Europe and the US …)
  - Unfortunately from the beginning of software development the productivity increase with software often outperformed the cost of low quality (except for safety critical systems); this allowed the industry to optimize profit vs. quality

What further progress could SWE make?

Your ideas?

Software Engineering - Advanced

- Topics [28]:
  - 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 [31]: Composition of design patterns
  - Jatoth et al. [32]: Literature Review on QoS-Aware Web Service Composition
  - Andreou, Papatheocharous [40]: Automated matching of component requirements
- New advances in formal methods:
  - Abrial [33][34]: Event-B method and toolset, industrial applied in
    - Railway engineering [35]
    - Real Time Operating System Memory Manager [36] (an excellent example of the application of Event-B)
  - Morales, Capel [37]: Model checking for critical systems

Software Engineering - Advanced

- Modeling
  - ThingML approach for IoT [29]
  - IoT Reference Architectures [30] and comparison
- Code generation
  - On-the fly for scientific computing [38]
  - Safety-critical avionics software [39]
- Simulation
  - Comparison of performance prediction methods [40]
- Etc., etc.

Software Engineering - Advanced

- Missing
  - Domain-specific and empirically confirmed standard sets of software quality properties
  - Domain-specific standard sets of a software components runtime parameters
  - E.g.:
    - Ressource metrics with respect to a reference platform/in a reference network
    - Correctness proven yes/no
    - …

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] ISO/IEC 25010 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Systems and software quality models

[2] Y. Zhang, X. Liu, Z. Liu and W. Li, "Development and Reconstitution of Software Quality Measurement and Evaluation Standards," *2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, Busan, Korea (South), 2018, pp. 380-384.

[3] ISO/IEC 25010 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of system and software product quality

[4] Y. Zhao, J. Gong, Y. Hu, Z. Liu and L. Cai, "Analysis of quality evaluation based on ISO/IEC SQuaRE series standards and its considerations," *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, Wuhan, China, 2017, pp. 245-250.

[5] J. M. Franco, F. Cerveira, R. Barbosa and M. Zenha-Rela, "Modeling the Failure Pathology of Software Components," *2016 12th International ACM SIGSOFT Conference on Quality of Software Architectures (QoSA)(QOSA)*, Venice, Italy, 2016, pp. 41-49

[6] C. Calero, A. Caro, M. Moraga and C. Moraga, "SQuaRE-Aligned Data Quality Model for Web Portals," *2009 9th International Conference on Quality Software (QSIC 2009)(QSIC)*, Jeju, 2009, pp. 117-122

[7] R. Lincke, W. Löwe and T. Gutzmann, "Software Quality Prediction Models Compared," *Quality Software, International Conference on(QSIC)*, Zhangjiajie, Hunan, China, 2010, pp. 82-91

References

[8] C. Paterson and R. Calinescu, "Accurate Analysis of Quality Properties of Software with Observation-Based Markov Chain Refinement," *2017 IEEE International Conference on Software Architecture (ICSA)*, Gothenburg, Sweden, 2017, pp. 121-130.

[9] E. Tang, X. Zhang, N. T. Muller, Z. Chen and X. Li, "Software Numerical Instability Detection and Diagnosis by Combining Stochastic and Infinite-Precision Testing," in *IEEE Transactions on Software Engineering*, vol. 43, no. 10, pp. 975-994, 2017.

[10] P. Rempel and P. Mader, "Preventing Defects: The Impact of Requirements Traceability Completeness on Software Quality," in *IEEE Transactions on Software Engineering*, vol. 43, no. 8, pp. 777-797, 2017.

[11] T. Besker, A. Martini and J. Bosch, "Impact of Architectural Technical Debt on Daily Software Development Work — A Survey of Software Practitioners," *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Vienna, Austria, 2017, pp. 278-287.

[12] J. Al Dallal and A. Abdin, "Empirical Evaluation of the Impact of Object-Oriented Code Refactoring on Quality Attributes: A Systematic Literature Review," in *IEEE Transactions on Software Engineering*, vol. 44, no. 1, pp. 44-69, 2018.

[13] N. Laranjeiro, S. N. Soydemir and J. Bernardino, "A Survey on Data Quality: Classifying Poor Data," *2015 IEEE 21st Pacific Rim International Symposium on Dependable Computing (PRDC)*, Zhangjiajie, China, 2015, pp. 179-188.

References

[14] P. Zhang, X. Zhou, W. Li and J. Gao, "A Survey on Quality Assurance Techniques for Big Data Applications," *2017 IEEE Third International Conference on Big Data Computing Service and Applications (BigDataService)*, Redwood City, USA, 2017, pp. 313-319.

[15] N. Nagappan, B. Murphy and V. Basili, "The influence of organizational structure on software quality," *2008 ACM/IEEE 30th International Conference on Software Engineering. ICSE'08(ICSE)*, Leipzig, 2009, pp. 521-530.

[16] F. A. Bianchi, A. Margara and M. Pezze, "A Survey of Recent Trends in Testing Concurrent Software Systems," in *IEEE Transactions on Software Engineering*, vol. 44, no. 8, pp. 747-783, 2018.

[17] D. Feitosa, A. Ampatzoglou, P. Avgeriou and E. Y. Nakagawa, "Investigating quality trade-offs in open source Critical Embedded Systems," *2015 11th International ACM SIGSOFT Conference on Quality of Software Architectures (QoSA)(QOSA)*, Montreal, QC, Canada, 2015, pp. 113-122.

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

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

[20] 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

References

[21] 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

[22] 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

[23] 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

[24] 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

[25] 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

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

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

[28] 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

References

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

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

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

[32] 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

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

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

[35] 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

[36] 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

[37] 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

References

[38] 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

[39] 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

[40] 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)*