Software & Systems Modeling (2018) 17:363–364
https://doi.org/10.1007/s10270-018-0674-5

EDITORIAL

Software engineering methods in other engineering disciplines

Jeff Gray[1] · **Bernhard Rumpe**[2]

Published online: 3 April 2018

Software engineers are often told from experts in systems development that they "should develop their software in the same systematic and predictable way as other engineers (e.g., mechanical engineers)".

If we trace Software Engineering back to the 1968 NATO Science Committee [1] conference in Garmisch, Germany, organized by F. L. Bauer, then the discipline of Software Engineering is approaching 50 years old. Yet, Software Engineering is not a discipline that always develops products using processes similar to traditional mechanical engineering methods. Software developers have their own portfolio of methods, ranging from heavyweight documentation-oriented approaches to the much more beloved agile and light-weight methods. Even though there is always potential for optimization, agile techniques have reached a level where software developers can produce reliable products using cost efficient, relatively predictable and controllable processes. What seems to be even more important is that the way of developing software is tightly connected to innovative processes that not only lead to novel ideas about how to implement software, but also to new ideas about potential features and services that can be integrated into software solutions. The tight coupling of software development and innovation is closely related to the strong connection between requirements elicitation and direct implementation in agile processes, where the same stakeholders are responsible for the innovations of development. Furthermore, the invention of new control and data structures in object-oriented development can drive the innovation of new features.

Companies in Silicon Valley have used this mood of innovation effectively. They have integrated software services into their core business model, which has led to innovative thinking and effective, agile development of products in other domains (e.g., medical services, autonomous driving, electric cars, finance). Traditional companies based on fundamental engineering processes may feel threatened by the radical change and fast rate of innovation that the new technologies offer. There are challenges in training classical engineers to adopt a new form of project organization, where the responsibilities are given to the developers much more than to the management hierarchies. However, this mindset is necessary to improve innovation. Traditional engineering-based companies, as well as other increasingly software-intensive companies, are now trying to catch up. Many good examples in these areas give us hope that change is possible.

In systems that require expertise from multiple areas of engineering, traditional engineers are becoming more open to software engineering practices. Furthermore, software development methodologies are being adopted more frequently into traditional engineering practice. In a recent and well-known German TV Show [2], a highly esteemed expert in mechanical and production engineering, Günther Schuh, was asked why he and his team were able to produce the "StreetScooter" electric car so quickly. Schuh said, "Because they have used computer science methods". In particular, Schuh mentioned agility as a key catalyst to the increased speed of development.

The portfolio of methods that software engineering has developed over the last two decades is effective and efficient in multiple engineering domains, allowing innovative products to be created at a quick pace. After 50 years, Software Engineering has found its toolset of methods, languages, concepts and techniques that allow software developers to create various forms of software products, services and embedded software, which in turn enables various new business concepts. One of our main problems now is to teach these practices in appropriate forms to many more computer scientists and other engineers, such that there are enough people available to deliver new innovative projects in the future.

✉ Bernhard Rumpe
bernhard.rumpe@sosym.org

Jeff Gray
jeff.gray@sosym.org

[1] University of Alabama, Tuscaloosa, AL, USA

[2] RWTH Aachen University, Aachen, Germany

1. Software Engineering: Report on a Conference Sponsored by the NATIO Science Committee, Garmisch, Germany, October 7–11, 1968. http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF.

2. Markus Lanz, February 2018, with Professor Günther Schuh. http://bit.ly/lanz-feb2018.

Content of this issue

In this issue, we have an expert's voice paper on "A logical approach to systems engineering artifacts: Semantic relationships and dependencies beyond traceability—from requirements to functional and architectural views" written by Manfred Broy. This contribution offers excellent insights into a method of developing distributed asynchronously communicating systems and how to precisely trace the high-level specifications of the requirements down well-architected and structured units of code.

In addition to one regular paper, this volume contains the Theme Section on "Performance Modelling and Engineering of Software and Systems" with Catalina Lladó and Kai Sachs as guest editors. This volume also contains the papers for the Special Section on "Business Process Modeling, Development and Support" with Selmin Nurcan and Rainer Schmidt as guest editors. Both sections contain six papers and a guest editorial describing their content.

The only regular paper is:

- "What can we learn from enterprise architecture models? An experiment comparing models and documents for capability development" by Ulrik Franke, Mika Cohen, and Johan Sigholm.