Software & Systems Engineering: Integrating Technology into Our Everyday Lives

Professor Manfred Broy
SOFTWARE & SYSTEMS ENGINEERING: INTEGRATING TECHNOLOGY INTO OUR EVERYDAY LIVES

Software is becoming an increasingly important element of our everyday lives – permeating many of the technologies we use regularly. As our software systems become increasingly sophisticated and ingrained into our lives, it is critical to our livelihoods that they remain both functionally correct and easy to use, even as their complexity increases. Professor Manfred Broy and his colleagues at the Technical University of Munich aim to ensure a smooth transition to a more software-dependent world, through new advances in the field of Software and Systems Engineering.

Software & Systems Engineering

As digital technologies rapidly improve, they are becoming increasingly integrated with the physical world, and therefore, our everyday lives. The key to the success of these technologies is the software that tells them how to run, and ultimately, determines their functionality. ‘Today, software is becoming the most important force in technology,’ says Dr Manfred Broy of the Technical University of Munich. ‘Therefore, it is key to understand and manage the classical tasks in Software and Systems Engineering.’

These classical tasks can be summarised by their aims to define real-world goals, and the behaviours and limitations of the software in realising them. Finally, it will be implemented by the command sequences that will see these goals achieved into physical hardware, through computer code. Once the software is implemented, Software and Systems Engineering continues to ensure that its specifications and real-world goals are being met, while managing its evolution over time, and between different systems. In recent years, these techniques have ensured that software continues to run reliably, minimising any potentially disastrous hitches.

A Need for Rapid Improvements

Despite significant improvements in Software and Systems Engineering, Dr Broy believes that they may not be happening fast enough. As the capabilities of new technologies grow in complexity, the software required to carry out tasks in the physical world is now itself growing increasingly large and complex. To ensure that these technologies are reliable for their many millions of users around the world, the capabilities of Software and Systems Engineering will, therefore, need to change with them. As Dr Broy explains, concerted efforts to develop these systems in a clean methodological way are now needed across a diverse range of current technologies.

‘There are a large variety of applications, including automotive systems, avionics systems, production automation, telecommunications, robotics, client-server systems, web-based systems and many more,’ he summarises. With so many of us using these technologies every day, Dr Broy now sees the necessity to push Software and Systems Engineering beyond its current capabilities as a problem that quickly needs addressing.
Adapting for Real-World Interaction

In the past, computer scientists involved in Software and Systems Engineering have widely succeeded in their goals to build system and software architectures, implement software using code, and verify the quality of software systems over time. Now, Dr Broy believes that the growing integration of software and physical users calls for more advanced Software and Systems Engineering systems that can capture this interaction. Ideally, such techniques would result in user and system interfaces that ensure technologies are easy to use, maintain, and move between different hardware, while being safe, secure and reliable.

‘Since software systems today are mostly embedded into cyber-physical systems as well as global networks, and interact with the physical world including users, it is most important to capture this interaction also in looking at key concepts related to the physical world,’ says Dr Broy. In their research, Dr Broy and his colleagues aim to achieve these properties by building a platform for Software and Systems Engineering, which supports this physical interaction, through the use of computer models that address several key issues.

Developing a System Model

Dr Broy and his team’s first step towards achieving this ease of interaction has been to build a reliable foundational model, onto which improvements in Software and Systems Engineering can be easily implemented. The ultimate aim of such a model is to ensure that user interfaces can convey complex systems of software, in ways that make them easy for everyday users to interact with. ‘The centre of the work is around a system model which supports all the previously-mentioned aspects by following the key principles of Software and Systems Engineering, such as interface abstraction, encapsulation, information hiding, modular composition, and flexible specification based on a concept of interface and system composition,’ says Dr Broy.

As well as this ease of interaction, the model needs to ensure that interfaces can reliably convey what the software is doing to their users, while maintaining simplicity. At the same time, the model addresses the needs of the developers. ‘My work encompasses different styles of modelling such as direct logical modelling of interface behaviour, and the modelling of interface behaviour by state machines,’ Dr Broy continues. ‘This leads into possibilities to model the key issues of systems.’ Ultimately, Dr Broy describes three aspects where these foundational models will be particularly important: creating improvements in system interfaces, devising ways to divide them into subsystems, and building sophisticated system architectures.

Improving External Interfaces

The first of these issues relate to how system functionalities can be improved, by ensuring ease of interaction with its users through external interfaces. ‘The function, or feature architecture, describes the functionality of a system in the form of its external interfaces, including the interface behaviour,’ Dr Broy says. In this context, ‘feature architecture’ refers to the high-level structures of computer code that underly the running of software. Just
like real architects, computer programmers need to fine-tune the details of their designs to ensure that their software runs smoothly.

As the only bridge between potentially highly complex software, and the users who likely have no knowledge of its inner workings, the functionality of an external interface is crucial to ensuring reliable system operations. The ‘feature architecture’ that Dr Broy describes, therefore, ensures that interfaces behave in such a way as to simply summarise to the user what the software is doing, while also ensuring that users have access to all of the capabilities of the software.

**Defining Logical Subsystems**

Secondly, Dr Broy’s model acknowledges how complex software needs to be broken down into less complex ‘subsystems’, making them easier to use for developers. ‘Logical subsystems or component architectures decompose a system into a number of subsystems or components which are composed to deliver the required functionality,’ he explains. From the cells in a beehive’s honeycomb to the modular parts of a space station, a wide variety of systems benefit from being broken down into easier-to-manage subsystems, and software is no exception.

If not carried out correctly, however, creating this kind of modular system could be detrimental to the overall system. Dr Broy and his colleagues address this issue by breaking systems down into modules that work together to achieve the goals of the software as a whole. Therefore, their model can turn sprawling software into sets of smaller, simpler systems – making them easier to convey to everyday users through external interfaces, while retaining their capabilities.

**Constructing Technical Architectures**

Finally, the subsystem architecture ensures that the sequences of events that define the running of the software are carried out without any major issues. ‘Technical architectures consist of hardware, deployment, and scheduling,’ Dr Broy explains. ‘In particular, they consider issues of computational complexity, performance, the technical realisation of such systems.’

Again, in comparison to the work of an architect, ‘technical architecture’ refers to the finely-tuned designs of software applications.

As the physical manifestation of the instructions carried out by software, hardware is a critical element to consider in Software and Systems Engineering. If the system is to function correctly, the software’s commands must be within the capabilities of the hardware. The team’s model, therefore, ensures that hardware runs smoothly during its interactions with users, while maintaining the function of all of the complex tasks required of it.

**Bridging the Gap**

Through developing these three key architectural views, Dr Broy believes his team’s work will result in improved principles in Software and Systems Engineering – forming a reliable, ubiquitous platform for developing new software. ‘The approach provides the theory for all these issues, and therefore forms the foundation of a comprehensive approach to the development process, the required methods, and tool support,’ he says.

Ultimately, these efforts could ensure that interactive technologies with ever-increasing complexity will continue to be both reliable and easy to use, as we become increasingly dependent on them in our everyday lives. In future research, Dr Broy and his colleagues will focus on improving their models further, with the ultimate goal of making them commercially viable. ‘The next steps will be to bring theory closer to practice; combining it with existing, more pragmatic modelling approaches,’ he concludes.

If the team achieves its goals, the increasingly complex technologies of the future will be able to reliably carry out their vital roles. Even as these systems become increasingly integrated into our everyday lives, future developments in Software and Systems engineering will ensure that complex software can continue to do its work behind the scenes as we interact with it.
Meet the researcher

Professor Manfred Broy
Department of Informatics
Technical University of Munich
Garching
Germany

Professor Manfred Broy received his PhD in Informatics in 1980 at the Technical University of Munich (TUM), Germany, for a thesis focussing on the transformation of programs running in parallel. In 1983, he founded the Faculty of Mathematics and Informatics at the University of Passau in Germany, where he worked as the dean for several years. In 1989, he moved back to TUM, where he founded the Department of Informatics. Here, he worked as a full professor until his retirement in 2015. Over the course of his career, Dr Broy has achieved numerous awards. In 1994, he received the Leibniz Prize from the German National Science Foundation (DFG) for his outstanding academic achievements, while in 2003, he obtained a Doctor Honoris Causa from the University of Passau. In 2007, he won the Konrad Zuse Medal, which is the highest award of the German Computer Science Society. Dr Broy is a member of the German Academy of Natural Sciences and the German Academy of Engineering Sciences. His main research interests involve the scientific modelling and development of powerful, software-intensive systems.

CONTACT
E: manfred.broy@mytum.de

FURTHER READING
