Current and Future Challenges of Software Engineering for Services and Applications

Sotiris Koussouris\textsuperscript{1}, Elisabetta Di Nitto\textsuperscript{2}

ICT and, in particular, software is more and more pervasive. It is affecting our lives, the services we can exploit, business, manufacturing, agriculture, health, and other fields in a way that could have never been even imagined a century ago.

Countries such as Taiwan, South Korea and the US spend a significant amount of their R&D investments in ICT, with Taiwan and South Korea mostly focusing on hardware, and the US having a balance between hardware and software. The situation in Europe varies from country to country, but in all cases ICT is significantly less prominent with about 20\% of R&D investments for France and Germany\textsuperscript{3}.

Software does not require complex machinery to be developed, it can be created on personal computers that today are accessible to almost all people in the society. This gives the impression that it can be developed by anyone with good technical skills and willing to learn some simple-to-use programming language. At the same time, its intangibility makes it invisible and, thus, suggests that it is only a minor part of the devices (information systems, factory automation, medical devices for diagnostic, mobile phones, sensors, ...) it is controlling, while, in many cases, it is a core part of it.

For these reasons, historically, there has been a tendency to direct investments and attention to the devices rather than to the software itself, and assume that software development and operation approaches and tools should have been defined within specific application domains, so to address specific problems pertinent to these domains. Also, in many cases, the developers of such approaches and tools as well as of the software itself were application domain specialists rather than software engineers. This is especially true for scientific software which is argued to be one of the main reasons why various software issues are attributed to this tendency\textsuperscript{4}.

As researchers in software engineering have been highlighting whenever possible, high quality software requires specific skills and the adoption of good and controlled development and operation practices. Software engineering has the mission to offer the right tools and methods to guide users in all activities connected to the lifecycle of software and services, through the usage of technologies and new paradigms, still ensuring productivity of processes and quality of software.

The last years advancements have led to an increasing automation of aspects such as testing, deployment, management of new software releases, and, at the same time, have allowed researchers and practitioners to identify new approaches for creating and operating software and services (think of DevOps as an example). However, we cannot stop researching as the problems we face show increasing complexity. In contexts such as Cloud, IoT, Big Data, Cyber-Physical Systems, the core part of software is aiming at creating the vital infrastructure and middleware layers required to enable information storage, transfer and transformation, and integration with other systems, while end-user applications are focusing on offering information and services to the users.

To design, build, manage and maintain such infrastructural software, and the way it interfaces with other components, we need to consolidate the software engineering discipline, which, despite the impressive achievements in the area of software technology, is probably one of the youngest scientific and technological disciplines with about 60 years of history.

The software engineering community needs to support this mission that will increase its ability to deal with a large number of challenges in other disciplines, from the achievement of a complete digitalisation of

\textsuperscript{1} National Technical University of Athens, skous@me.com
\textsuperscript{2} Politecnico di Milano, elisabetta.dinitto@polimi.it
Short Overview Of Challenges In Software Engineering

The challenges identified encompass the process and technological aspects described below. A more detailed description can be found in the next section.

In the context of PROCESS, METHODOLOGIES AND PRODUCTIVITY, existing concepts need to be redefined for meeting the current needs of the industry. A new notion of productivity should be defined, where “lines of code” is not anymore the right measure of productivity, but software is measured in terms of its other qualities, usability, reliability, scalability. New possibilities to easily gather user feedback and monitoring information have the potential to enable an informed evolution of software while shorter development cycles call for novel software production methodologies to actually enable controlled management of such short development cycles. With DevOps there is also a need to shift deployment decisions and resource management from the deployment phase to the design phase of software engineering, making efficient use of resources and supporting architecture level analysis, optimisation of deployment decisions, as well as automated placement and orchestration of applications/services (e.g. specification of novel software development paradigms based on micro-services) and adopting infrastructure-as-a-code approaches for eliminating the needs for configuration placement and management over programmable infrastructures, with an emphasis on their utilisation from small companies.

Software is also the driving force behind the CPS and IoT paradigms, and their further evolution is heavily linked to the ability of software to be both dependable and adaptable to real time changes, thus enabling different APPLICATION CONTEXTS. The main challenges raised by IoT-enabled CPS include the development of models, methods and design tools for IoT/CPS-enabled applications going beyond formal methods research to create abstractions and formalisms for constructing and reasoning about systems with diverse and more difficult-to-characterize components. CPS also need novel methods for software Adaptability, Scalability and Maintainability, as such systems need to be continuously modified and maintained to meet changing requirements during runtime. In this context, the new ingredients are likely going to be run time adaptation of software Quality Assurance in large scale open CPS environments able to deal with uncertainty and variability at the same time, software-awareness of hardware to ensure Web-scale performance, flexibility and agility and meet the “software-defined anything paradigm.

As the suggest above, and having in mind the different application context, there is also a need to further research into DESIGN PATTERNS DEVELOPMENT FOR A SYSTEMS OF SYSTEMS APPROACH. New patterns at the architectural level describing the obligations/constraints to be fulfilled by the system in which the software is running, and to validate and standardise them are needed and methods on how to apply them into a dynamic, ever-changing context environments. As such, issues such as frame of references, unifying lexicons, visualisations, design architecture and interoperability, modelling languages, tools integration and simulation and analysis should be tackled.

Such patterns will allow software to reach a better level of quality. At present, the problem of offering proper QUALITY GUARANTEES has to be reconsidered to cope with the emerging trends in software engineering and accelerate the adoption rates of novel methods such as the agile delivery. The rapid growth in the last years of agile delivery methods in the context of DevOps call for research to increase the anti-fragility of systems, reduce the meantime-to-restore-service (MTRS), and develop accelerated methodologies to test quality through testbeds. In parallel, although Big Data offers the ability to capture large amounts of monitoring data on the behaviour of an application, limited progress has been achieved in developing feedback analysis tools, thus further research is envisaged in the architectural level, in the ability to pinpoint specific root causes of performance degradation in the application code, and in the application of machine learning methods for quality engineering. There is also a shortage of reference quality benchmarks for code and extra-functional properties in many classes of applications and domains.

One other core aspect of research, directly linked to any software engineering activity is that of REQUIREMENTS ENGINEERING. Going away from monolithic and stand-alone applications and adopting a Digital Single Market and Connected-world mentality increases complexity of knowledge capturing and representation. New devices, services and even individuals become part of a software-powered ecosystem and flexibility, constant evolution and interconnection contradicts current requirement engineering
outputs, as existing approaches do not account for dynamicity of use and unknown requirements. There is the need for a radically divergent approach to capture emerging behaviour from systems and users. Emerging technologies and trends are shedding light on potential research topics such as multichannel big data analytics for requirements elicitation from large scale sites (like smart-city infrastructures which blend humans, machines and generally system characteristics and behaviour), novel methods for user engagement towards directly extracting requirements, privacy respective indirect requirements extraction paradigms exploiting context-awareness of individuals independently on the usage of a specific software, HMI interface types taking into account CPS and new technologies that blend human and computer interactions and decisions, different kind of logics (both rational and behavioural), interconnection and interoperability with next-of-kin and other unrelated (at first sight) systems of a greater ecosystem, placement of requirements into production schedule dictates and relation with emerging business needs, unanimous description conventions and abstraction representation levels.

**PRIVACY AND SECURITY** at design time as well as runtime of the software is another important aspect that should be tackled to comply with the evolution of software development methodologies (e.g. microservices based software engineering approaches) along with the placement of applications over virtualised environments in multiple formats (e.g. virtual machines, containers, unikernels), the threat and vulnerabilities landscape is continuously expanding. Special care with regards to privacy and security has to be given in complex distributed systems that in many cases have to handle big data volumes in a distributed way. Challenges include identification contextual systems patterns related to privacy leaking code snippets, secure computation of data structures, approaches for establishing optimality of encryption levels, continuous source code assessment at design time as well as vulnerability assessment of the developed applications, secure packaging and placement mechanisms of the developed applications over programmable infrastructure, orchestration mechanisms supporting the secure and efficient policy-aware management of services and applications, real-time risk identification and assessment techniques along with the triggering of the appropriate mitigation actions, security and privacy mechanisms focused on distributed and big data applications.

As datasets handled by software are constantly increasing, apart from supplying novel algorithms, new system architectures and software infrastructures able to cope with the 5Vs of BIG DATA FOR AND OUT OF SOFTWARE ENGINEERING, it’s high time for software itself to benefit from the intelligence extracted from large sets of information such as software source code, commits and forks, bugs, warnings and notifications, issues from backtracking systems, logs of any kind, commits, demographics, coding patterns, requirements, user behaviours, user profiles, etc. Research challenges for software engineering in this direction include novel tools employing techniques of machine learning and data mining to reveal hidden knowledge aspects and extract information from sensor-based architectures, excavating knowledge which is impossible for humans to dig out, but is necessary to be brought into human attention and affection for improving software qualities, studying the evolution / discontinuation of application frameworks, open source components, analysis of user trends and preferences and behaviour with systems to better understanding users’ needs, tools and methods for identifying feature and performance improvement opportunities, identifying root causes of failures and system halts based on log files (massively big (>>GBs) or lightning-fast updating) coming from various complex distributed systems and infrastructures, insights collected at runtime on symptoms and context changes triggering adaptations, and perform predictive and prescriptive analytics for proactive planning and preparation of adaptation actions.

Finally, there is a continuous need towards accelerating OPEN SOURCE SOFTWARE INNOVATION. Many projects lack proper community engagement and management structures, quality assurance, and a vision on how to contribute to the European open digital market. OSS governance includes a number of technical challenges related to software engineering and production processes, including methodologies and tool support for the detection and disposition of contradictions, ambiguities, and gaps in requirements specifications, decoupled architectures and production processes based on fault defensive and tolerant programming styles for distributed developers teams with different skill sets, interests and motivations, methodologies and tools for impact analyses of code additions and modifications. Furthermore, OSS production processes also include organisational challenges that have to be met by an interdisciplinary approach aiming at the creation and management of communities of code contributors, reviewers, testers, first level users, etc. and a comprehensive development and communication approach combining existing tools under a set of common, formalised set of methodologies.