# Current and Future Challenges of Software Engineering for Services and Applications

Casale, G.; Chesta, C.; Deussen, P.; Di Nitto, E.; Gouvas, P.; Koussouris, S.; Stankovski, V.; Symeonidis, A.; Vlassiou, V.; Zafeiropoulos, A.; Zhao, Z.

**DOI**
10.1016/j.procs.2016.08.278

**Publication date**
2016

**Document Version**
Final published version

**Published in**
Procedia Computer Science

**License**
CC BY-NC-ND

Link to publication

**Citation for published version (APA):**
Casale, G., Chesta, C., Deussen, P., Di Nitto, E., Gouvas, P., Koussouris, S., Stankovski, V., Symeonidis, A., Vlassiou, V., Zafeiropoulos, A., & Zhao, Z. (2016). Current and Future Challenges of Software Engineering for Services and Applications. *Procedia Computer Science*, *97*, 34–42. https://doi.org/10.1016/j.procs.2016.08.278

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 97 (2016) 34 – 42

Procedia Computer Science

CLOUD FORWARD: From Distributed to Complete Computing, CF2016, 18-20 October 2016, Madrid, Spain

# Current and Future Challenges of Software Engineering for Services and Applications

Giuliano Casale[a], Cristina Chesta[a], Peter Deussen[a], Elisabetta Di Nitto [a,]*, Panagiotis Gouvas[a], Sotiris Koussouris[a], Vlado Stankovski[a], Andreas Symeonidis[a], Vlassis Vlassiou[a], Anastasios Zafeiropoulos[a], Zhiming Zhao[a]

[a]*Cluster of Europen Projects on Software Engineering for Services and Applications*
*https://eucloudclusters.wordpress.com/software-engineering-for-services-and-applications/*

**Abstract**

ICT (Information and Communication Technology) and, in particular, software is more and more pervasive and it cannot be considered anymore as a minor element of a complex systems. In domains like cloud, big data, IoT (Internet of Things), CPS (Cyber-Physical Systems) it is the core element. 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. This paper summarizes the challenges that the Software Engineering for Services and Applications (SE4SA) cluster is considering as relevant.

*Keywords:* Software; Services; Research Challenges; Collaboration; Software Development

## 1. Motivation

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

* Corresponding authors.
*E-mail addresses:* elisabetta.dinitto@polimi.it, skous@epu.ntua.gr

doi:10.1016/j.procs.2016.08.278

century ago. Countries such as Taiwan, South Korea and USA spend a significant amount of their R&D investments in ICT, with Taiwan (more than 70%) and South Korea (more than 50%) mostly focusing on hardware, and USA (more than 40%) on both hardware and software. According to Shackelford & Jankowski[1], 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.

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 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 to assume that software development and operation approaches and tools should have been defined within specific application domains, so to address specific problems peculiar of that 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, as discussed on Nature, where Merali[2] points out at a number of issues generated by this tendency.

Retrospectively, we can see that the lack of mature software engineering expertise or practices have caused software bugs that, in turn, have determined loss of significant quantities of money in the best cases. The last prominent case has happened very recently, in February 2016, when the Japanise Hitomi spacecraft broke off for a problem that was caused by a number of software failures[3].

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 (performance, availability, evolvability, reliability, …). In 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[4] as an example). However, we cannot stop researching as the problems we face show increasing complexity.

In contexts such as Cloud, Internet of Things (IoT), Big Data, Cyber-Physical Systems (CPS), the core part of software is aiming at creating the infrastructure and middleware layers needed to enable execution of services, storage, transfer and transformation of information, and integration with other systems, while end-user applications are focusing on offering information and services to the users. In these contexts, such core infrastructural software has to offer important guarantees in terms of Quality of Service (QoS) and correctness. Moreover, it has to deal with an open world in which the resources it is exploiting as well as the other software and devices it is interacting with, change for many reasons, like failures, changing interfaces and implementations, changing requirements, etc.

To build and manage such infrastructural software, and the way it is interfaced 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 community needs to support this mission that will increase also our ability to deal with a large number of challenges in other disciplines, from the achievement of a complete digitalization of public administrations and businesses to the actual development of smart cities, to the development of reliable software for science, to the discovery of new fields that we cannot imagine today.

Within this context, the cluster on Software Engineering for Services and Applications (SE4SA) is a forum where European projects funded by the European research programmes collaborate to identify synergies, possibilities of collaboration and new challenges to be tackled in future initiatives. The aim of this paper is to provide an overview of the areas that are currently covered by the projects belonging to the cluster (see Section 2) and to propose new challenges for future research (Section 3). The currently covered areas have been identified by running a survey within the cluster, while the new challenges have been identified as a result of discussions developed by the cluster and involving also external stakeholders during ICT 2015 and NetFutures 2016.

Table 1. Projects that have filled in the questionnaire.

| Project | website | Project | website |
|---|---|---|---|
| ALIGNED | http://aligned-project.eu/ | Envisage | http://www.envisage-project.eu/ |
| AppHub | https://www.apphub.eu.com/ | HyVar | http://www.hyvar-project.eu/hyvar/ |
| ARCADIA | http://arcadia-framework.eu/ | MODAClouds | http://www.modaclouds.eu/ |
| ARTIST | http://www.artist-project.eu/ | Prowess | http://www.prowessproject.eu/ |
| CloudTeams | http://www.cloudteams.eu/ | RISCOSS | http://www.riscoss.eu/bin/view/Main/ |
| DICE | http://www.dice-h2020.eu/ | S-CASE | http://www.scasefp7.eu/ |
| ENTICE | http://www.entice-project.eu/ | SWITCH | http://www.switchproject.eu/ |

## 2. Challenges Dealt by the Current European Projects in Software Engineering

Taking into account the evolving landscape of the software engineering challenges and approaches, a reporting activity of the current challenges faced by existing EU-funded research projects has been realised by the cluster of European projects on Software Engineering for Services and Applications. The results of this activity are shortly presented in this section. Upon the identification of the set of tackled challenges from the participating projects in the cluster, a taxonomy of these challenges is produced through their clustering in the following categories: software design challenges, software placement/implementation challenges, orchestration middleware challenges, software quality challenges and services/applications lifecycle management challenges.

The *software design challenges* regard the design and development of novel software engineering approaches, taking into account the transition from monolithic applications to applications developed based on the adoption of microservices-based paradigms and the need for novel approaches that facilitate service/applications composition.

The *software placement/implementation challenges* concern the adoption of novel virtualisation approaches and the development of software that can take advantage of them (e.g. IoT based software in the form of containers, distributed applications consisted of microservices packaged as unikernels) and the associated security, distribution and delivery challenges.

The *orchestration middleware challenges* regard the need for design and development of orchestration mechanisms able to facilitate optimal placement and execution of applications over programmable infrastructure, taking into account the programmability of the infrastructure as well as the reconfigurability of the applications' execution context.

The *software quality challenges* refer to the need for adoption of approaches that will facilitate development of qualitative software, such as collaboration driven software development and testing processes ensuring interoperability and user acceptance, e.g., introduction of novel software engineering tools, adoption of open-source solutions, quality assessment, revolutionary methods for collecting feedback prior - during and after the deployment, evolution of a validation and verification culture for any software related product or service, utilisation of available (customer) data for improving assumptions during design/development, cloud services interoperability frameworks.

The *services/applications lifecycle management challenges* refer mainly to the need for interconnection of software engineering and DevOps functionalities/approaches and the support of continuous deployment approaches (e.g., tools for automated or semi-automated deployment of applications taking into account the existence of several components along with their requirements/constraints).

The aforementioned challenges categorization has been broken down into a detailed set of challenges per category, as explained in the following subsections. Based on the provided set of challenges per category, a campaign for collection of feedback from the projects participating at the cluster of European projects on Software Engineering for Services and Applications has been realised. An online questionnaire has been prepared and filled in for this purpose by partners of the projects in Table 1.

Feedback was collected for the challenges tackled per project and the way that they are going to be addressed, the importance given on the tackling of each category of challenges per project, as well as relevant future trends in software engineering. The importance given by the aforementioned projects to the tackling of the challenges per category is depicted at Figure 1. The highest rates appear at the software quality, software design and lifecycle management categories, where various approaches are under design and development. The software implementation and orchestration middleware categories can be considered as emerging ones in terms of challenges, taking into account the continuous evolution of virtualisation technologies and reactive software development paradigms. In the following, we provide detailed information regarding the challenges identified per category.

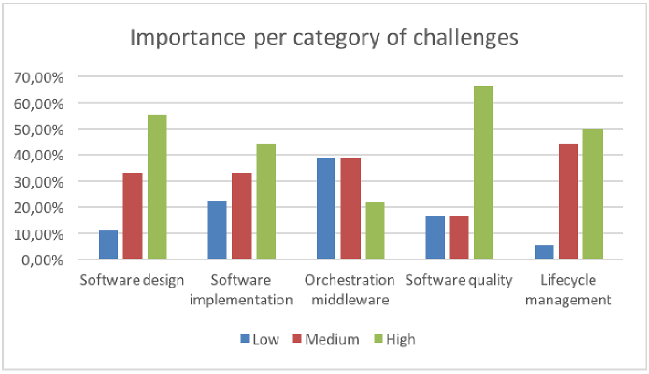

Fig. 1. Importance per category of challenges.

## 2.1. Software Design

In the software design challenges, the transition from application development approaches to application composition approaches is considered crucial. Towards this direction, the support of software reuse through the development of software based on microservices combined with software composition frameworks has to be realised. Actually, a shifting from code-heavy monolithic applications to smaller, self-contained microservices is realised. Applications with a microservice architecture consist of a set of narrowly focused, independently deployable services. Such services can be combined in order to produce service chaining graphs and support advanced functionalities. The deployed microservices have to be validated based on a unified representation model with regards to their characteristics and interconnection interfaces.

It is also stated that support has to be provided for software development and evolution by design, by applying reusable design patterns, separation of concerns and high level modelling, supporting either the discovery and understanding of complex software systems, and their re-architecture for modernization and evolution.

In many cases, several approaches are considering the usage of annotations in software level that can be interpreted during execution time. Based on these, it is possible to build software repositories that annotate software based on its functional and non-functional characteristics. Such annotations may denote -among others- preferences with regards to the parallel execution of code, scalability patterns to be followed, constraints, preferences with regards to virtualization technologies to be used.

Within the software design challenges are included also challenges for the development of self-adaptive systems that support responsiveness, fault tolerance, self-healing and variability characteristics. It is stated that the developed software components have to be responsive-by-design to the changes in the operational environment (e.g. by adopting “let-it-crash” models). The developed systems have to support adaptability characteristics, including elasticity and (horizontal and vertical) scalability characteristics. Elasticity and horizontal scalability characteristics have to be supported, possibly through the design and application of stateless mechanisms. Furthermore, support has to be provided to concurrency, parallelism and distributed functionalities.

During software design, we have also to take into account the variability of highly distributed applications in heterogeneous environments. The spatial variability, which captures different possibilities for configuring the distributed application instances before they are deployed, is very large. The temporal variability, which captures different possibilities for runtime reconfiguration, is equally large and the computation of software upgrades is complex; in particular, the reconfiguration may depend on the environment of the application instance, as reflected in sensor data. It should be noted that almost half of the projects address the part of the challenges that has to do with the design and development of reusable design patterns, while one third of the projects tackle the challenges related with the design of self-adaptive software that supports scalability and elasticity characteristics.

### *2.2. Software Implementation and Orchestration Middleware*

In the software placement/implementation and orchestration challenges, the exploitation of programmability aspects of the underlying infrastructure, for instance, Software Defined Networking (SDN) and Network Function Virtualization (NFV), is considered crucial. Also, there is a set of software-engineering challenges for adopting DevOps practices. The objective is to deploy applications over programmable infrastructure in an optimal way in terms of software performance as well as high-level services providers' policies. Thus, the provided software should be modeled in a way that provides flexibility to DevOps users (or automated deployment tools) to deploy it in an optimal manner. This applies mostly to applications with high distributed/concurrency/parallelism notion. Furthermore, effort is allocated in providing cloud-based services and tools for rapid software prototyping, in order to boost software productivity and reduce software development costs, as well as in the adoption of mechanisms for developing efficient software that takes into account big data management and high computational requirements.

Towards this direction, data locality, data volatility and time criticality issues drive the definition of a set of requirements that have to be fulfilled. Furthermore, it is important to be able to manage ready-to-run workloads designed for physical, virtual and cloud environments based on the usage of unique templates for all private, public or hybrid cloud environments. From the aforementioned challenges in this category, the most popular ones regard the need for provision of cloud-based services and tools for rapid software prototyping and the need for management of the complexity of large software and data-intensive systems.

### *2.3. Software Quality*

Regarding the software quality challenges, focus is given on improving the quality and acceptability of cloud based services through novel, collaboration-driven requirements elicitation and analysis, software validation and verification methods, tools and frameworks including all actors of the value chain, from software engineers and product manager to end-users. Effort is allocated towards the specification of mechanisms targeted at improvement of trust, transparency and interoperability of cloud-based services through the introduction of methods that allow (self/federated)-certification of outputs based on unanimously agreed procedures and standards.

There is also a need for approaches able to guide software quality evolution. Such approaches aim to drive the early design stages of the application development and offer proper quality engineering tools, as well as follow best practices in open-source software development, governance, licensing, testing, standards, etc. in order to help produce quality software that is easy to use. Software quality may be also significantly improved through the exploitation of massive online user feedback in requirements engineering. End-users are increasingly providing feedback through reviews, ratings and comments in online forums, app stores and social networks.

While this represents an opportunity to involve end-users in software evolution processes aiming at improving the quality of the offered software applications and services, it calls for methods and tools for gathering end-user feedback that need to scale and enable automated analysis of feedback and contextual data to support requirements analysts, system architects, developers and project managers in decision-making tasks. Towards this direction, another challenge is to integrate anonymised and privacy-respecting customer's activity data as early feedback for software design and development activities, coming not only from interaction with the software itself, but mainly from other activities as to better understands customers' contexts. Based on the data collected by the projects participating in the cluster activities, one third of the projects are tackling challenges related to the design of proper quality engineering tools for software quality evolution while lot of them (~28%) are also working towards improving the quality and acceptability of cloud based services through novel, collaboration-driven requirements elicitation and analysis, software validation and verification methods.

### *2.4. Lifecycle Management*

Finally, regarding the services/applications lifecycle management challenges focus is given on the adoption of model-based development techniques (more than half of the projects tackling such aspects) and of DevOps practices (one out of four of the projects).

Managing the development complexity and risks in both design and runtime phases is considered crucial, in order to increase QoS, reduce the time needed to move new releases in the operation environment and enable the constitution of new and more effective cooperation processes between the development and the operation team. The adoption of model-based development techniques covering the different phases of the software development life cycle, including automation on code synthesis, smell and anti-pattern detection, variability modeling in software product lines and software simulation techniques is also promoted for increasing productivity and quality of software.

Finally, efforts are also allocated towards the design of new software engineering, testing and deployment methods driven by the "security-by-design" perspective in order to respond to long-standing and emerging security and vulnerability threats that jeopardise the value of critical cloud based services.

## 3. Future Challenges in Software Engineering

Although, as indicated in the previous section, lots of challenges are tackled today at various levels, new challenges emerge as technology progresses and new concepts arise, while existing challenges take new turns as well, and further research and innovation transfer activities are necessary in order to be proactive and solve problems efficiently and effectively. In this section we highlight the challenges that the cluster has identified as the result of: i) a very lively discussion panel at ICT 2015, ii) workshops at Cloud Forward 2015 and NetFutures 2016 and iii) various internal meetings among cluster members.

### *3.1. Process, Methodologies and Productivity*

In the context of process, methodologies and productivity, existing concepts need to be redefined for meeting the current needs of the industry. Software process is a well-investigated research area, but nowadays there are several new advancements in technology and practice that introduce significant changes in this aspect. 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[4], making efficient use of resources and supporting architecture level analysis, optimization 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 utilization from small companies.

### *3.2. Application Contexts*

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 enable 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. Moreover, the needs brought by CPS for novel methods of software Adaptability, Scalability and Maintainability[5], which are not envisaged at design time as such systems, need to be continually modified and maintained to meet changing requirements, 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.

### *3.3. Design Patterns Development for a Systems of Systems Approach*

As 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 standardize them are needed and methods on how to apply them into a dynamic, ever-changing context environments[6]. 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.

### *3.4. Quality Guarantees*

Design patterns, as discussed previously, will allow software to reach a better level of quality. However, they alone are not enough. The rapid growth in the last years of agile delivery methods in the context of DevOps, as well as the need to reduce the development time as much as possible, call for research approaches that can increase the anti-fragility of systems, reduce the meantime-to-restore-service (MTRS), and develop accelerated methodologies to test quality through staging and canary 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 to quality engineering[7]. Last but not least, there is a shortage of standardized reference quality benchmarks for code and extra-functional properties in many classes of applications and domains.

### *3.5. Requirement Engineering*

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[8] (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, Human-Machine 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.

### *3.6. Privacy and Security by Design*

Privacy and security at design time as well as runtime of 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). These issues have become highly critical since the 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 the identification of 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[9], 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. Special emphasis should be given in the topic of security and privacy by design software engineering approaches that contribute in the generation of software artefacts that can operated in multi-IaaS environment with increased security characteristics.

### *3.7. Big Data for Software Engineering*

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, 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[10], 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, as a great proportion of the software developed today is of Open Source, 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[11]. 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, formalized set of methodologies.

## 4. Conclusion

With this paper we have provided an overview of the hot topics European research projects in the area of software engineering are addressing today and of the challenges that they see in the medium term. We do hope that this information will help in shaping the objectives of future projects in the area.

For what concerns our future research agenda, we plan to complete the mapping of the areas covered by current projects relevant to software engineering and validate the challenges that we have identified by extending the discussion to a larger community in the area.

## Acknowledgements

The work of authors is funded by the European Commission under the grant agreements of the corresponding projects.

## References

1. Shackelford, B., Jankowski, J. (2016). *Information and Communication Technologies Industries Account for $133 Billion of Business R&D Performance in the United States in 2013.* National Center for Science and Engineering Statistics. NSF.
2. Merali, Z. (2010, 10 13). *Computational science: … Error.* Retrieved from Nature: http://www.nature.com/news/2010/101013/full/467775a.html
3. Witze, A. (2016, 04 28). *Software error doomed Japanese Hitomi spacecraft.* Retrieved from Nature: http://www.nature.com/news/software-error-doomed-japanese-hitomi-spacecraft-1.19835.
4. Ebert, C. Gallardo, G., Hernantes, J., Serrano, N.(2016). DevOps, *IEEE Software*, vol.33, no. 3, pp. 94-100, doi:10.1109/MS.2016.68.
5. Gaaloul, W., Ezpeleta, J., Zhou, Z., Barhamgi, M., (2015) CPS 2015 Track Report: Cyber Physical Society", *WETICE*, 2015, 2015 IEEE 24th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE).
6. Lano, K., Kolahdouz-Rahimi, S., (2014) Model-Transformation Design Patterns", *IEEE Transactions on Software Engineering*, vol.40, no. 12, pp. 1224-1259, Dec. 2014, doi:10.1109/TSE.2014.2354344.
7. Zhao, Z., Martin, P., Wang, J., Taal, A., Jones, A., Taylor, I., Stankovski, V., Garcia Vega, I., Suciu, G., Ulisses, A., Cees de Laat, C. (2015), *Developing and operating time critical applications in clouds: the state of the art and the SWITCH approach,* In the proceedings of HOLACONF - Cloud Forward: From Distributed to Complete Computing, Pisa, Italy. Elsevier, Procedia Computer Science, Vol (68) page (17-28).
8. Lencastre, M. (2014), Foreword of the Thematic Track: Quality in ICT Requirements Engineering", *QUATIC*, 2014, 2014 9th International Conference on the Quality of Information and Communications Technology (QUATIC).
9. Bertino, E., Samanthula, B. (2014) Security with privacy - A research agenda, *COLLABORATECOM*, 2014, 2014 International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom).
10. Gorton, I., Bener, A., Mockus, A (2016) Software Engineering for Big Data Systems, *IEEE Software,* vol.33, no. 2, pp. 32-35, Mar.-Apr. 2016, doi:10.1109/MS.2016.47
11. Nagappan, M., Mirakhorli, M. (2015) Big(ger) Data in Software Engineering, *ICSE*, 2015, 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE) 2015, pp. 957-958, doi:10.1109/ICSE.2015.305.