Introduction to the Special issue on Engineering Context-aware software systems

Domenico Amalfitano

University of Naples Federico II

domenico.amalfitano@unina.it

Santiago Matalonga

University of the West of Scotland

santiago.matalonga@uws.ac.uk

Guilherme Horta Travassos

Federal University of Rio de Janeiro

IST Associate Editor

ght@cos.ufrj.br

Introduction

Context-awareness is "a dynamic property of a system that can evolutionarily affect the software system's overall behavior in the interaction between the actor and the computer"[1]. In turn, context is "any piece of information that may be used to characterize an entity's situation (logical and physical objects present in the systems´ environment) and the relations that are relevant to the actor computer interaction." (based on[1]). Therefore, Context-aware software systems can identify changes in the logical or physical environment (i.e., context) and adapt their behavior to provide better service to the actor.

Context-aware software systems (CASS) are now mainstream. In less than a year, the hype of the press went from CEOs arguing that "driverless trucks will never happen"[2] to the first successful coast-to-coast driverless truck journey in the US[3]. The hype of context-aware software systems is well deserved. However, we argue that the engineering methods have not yet been developed to ensure the safe and secure delivery of context-aware software systems. In particular, since 2015, we have been studying the availability of testing technologies to support the testing of a context-aware software system[4]. At the time, our main finding was that none of those proposals accept that context varies, and as such, that it should change during the testing process. Extending this into the whole life cycle process is needed to engineer context-aware software systems, and we still believe there is a need for more evidence-based research to

understand how to accommodate context in all lifecycle phases.

Notwithstanding, there is circumstantial evidence that the industry is developing, testing, and deploying context-aware software systems (self-driving cars are just one of the possible examples.) These efforts, usually led by the Big Tech (like Tesla, Uber, Amazon, among others), do not seem to have found an outlet in peer-reviewed academic literature. Therefore, we proposed a venue to host industrial reports of the engineering of context-aware software systems, expecting it to bridge the evidence gap in the technical literature.

To summarise, this special issue aims at identifying industrial practices that have yet to be published, or studied, in the mainstream academic literature—acknowledging that both the industry and academia are currently approaching the domain with practices and methodologies that can be of interest to computer scientists, software engineers, and practitioners.

In this special issue

The papers included in this special issue cover a range of topics within the engineering of CASS. However, they also uncover that the field is ripe and calls for more evidence-based research. The papers in this special issue offer complementing context views. There is a clear overlap between the context-awareness and self-adapting systems, as described in paper two. Furthermore, some areas of the software development lifecycle have not been covered by the papers in this issue (for instance, Requirements Engineering or Maintenance). In short, we are delighted to introduce these papers. Their evidence-based nature introduces important aspects to take care of when engineering context-aware software systems.

The paper "Context-Oriented Behavioral Programming" introduces a programming paradigm that accommodates context. It is a novel paradigm for developing context-aware software systems that extend behavioral programming by allowing b-threads to share contextual information. The work formally describes the proposed paradigm and presents conceptual motivations for adding context-oriented b-threads to a behavioral-programing paradigm and concrete implementations of increased complexity to show the feasibility of the approach. Arguably, the most complex of this is the demonstration of the approach for a smart building application. The examples show how the context-oriented b-threads can serve as orchestrators for the rule-based Behavioural programming threats. This programming paradigm can potentially simplify the development of context-aware software systems since the developer can focus on known rules and effects of context and behaviors without comprehensive details about all possible execution paths.

The "Multifaceted infrastructure for Self-Adaptive IoT Systems" presents an integrated evaluation of a set of tools aimed at supporting developers with the production of self-adaptive context-aware software systems. It is noteworthy that this paper ties in three coordinated research efforts within the Engineering of Context-aware software systems. With CoAP-CTX[5], the authors aim to facilitate service discovery in IoT self-adaptive systems. Next, with SUCCEEd[6], researchers aim to support self-adaptation by providing a framework that supports the specification of strategies and rules. Complementing with LoCCAM[7], authors aim to support context adaptation for context-aware software systems. This paper provides an integrated evaluation of the three components in the context of a smart building self-adaptive context-aware software system. Their results show a reduction in the software system's coupling and complexity at the cost of increased size, with no significant increase, and at times reduction, of the overall computation time for achieving adaptations.

The "Runtime Testing of Context-Aware Variability in Adaptive Systems" presents a tool for runtime

testing of adaptive systems (RETAkE). Retake is embedded into the application at deployment time to monitor the DAS adaptive capabilities. RETAkE mainly focuses on the testing of the context-aware variability of the system during its execution. It uses runtime information to support the verification of the system adaptation. The application developer must embed RETAkE instructions (in the form of JAVA annotations) to notify Retake Engine of the places in the application that will be monitored (and re-executed at runtime). After deployment, the RETAkE engine, which has to be deployed with the application, monitors the SUT perception of the context and generates a test sequence. Runtime testing consists of injecting data in the system under test to guide the adaptation and evaluating the produced output. Each test case needs one entire execution of the adaptation loop. Authors validate RETAkE with two real-life examples. Their validation aims at assessing the overhead cost of RETAkE in the system.

Acknowledgments

We thank and appreciate the 24 anonymous reviewers who contributed to the success of this special issue. We can not thank them enough for their commitment and the quality of their work. In particular, through this very challenging 2020.

References

[1] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, “Towards a Better Understanding of Context and Context-Awareness,” in *Handheld and Ubiquitous Computing*, vol. 1707, H.-W. Gellersen, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 304–307. doi: 10.1007/3-540-48157-5_29

[2] “Executives: Trucks likely will never be driverless,” Feb. 12, 2018. https://www.trucker.com/technology/executives-trucks-likely-will-never-be-driverless (accessed Dec. 13, 2020).

[3] “Ex Google engineer completes 3,000 mile coast-to-coast journey in driverless car,” Dec. 18, 2018. https://www.telegraph.co.uk/technology/2018/12/18/ex-google-engineer-completes-3000-mile-coast-to-coast-journey/ (accessed Dec. 13, 2020).

[4] S. Matalonga, F. Rodrigues, and G. H. Travassos, “Characterizing testing methods for context-aware software systems: Results from a quasi-systematic literature review,” *J. Syst. Softw.*, vol. 131, pp. 1–21, Sep. 2017, doi: 10.1016/j.jss.2017.05.048.

[5] F. M. Barreto, W. Viana, M. E. F. Maia, and R. M. Andrade, “CoAP-CTX: Extensão sensível ao contexto para descoberta de objetos inteligentes em internet das coisas,” presented at the XXXV SBRC 2017 - Brazilian Symposium on Computer Networks and Distributed Systems, 2017.

[6] B. R. A. A. Junior, R. M. C. Andrade, M. E. F. Maia, and T. P. Nogueira, “SUCCEEd: Support Mechanism for Creating and Executing Workflows for Decoupled SAS in IoT,” in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, Jul. 2018, vol. 02, pp. 738–743, doi: 10.1109/COMPSAC.2018.10329.

[7] M. E. F. Maia, A. Fonteles, B. Neto, R. Gadelha, W. Viana, and R. M. C. Andrade, “LOCCAM - loosely coupled context acquisition middleware,” in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, Coimbra, Portugal, Mar. 2013, pp. 534–541, doi: 10.1145/2480362.2480465.