Continuous Experimentation on Cyber-Physical Systems: Challenges and Opportunities

Federico Giaimo, Hang Yin, Christian Berger, Ivica Crnkovic
Chalmers | University of Gothenburg
Department of Computer Science and Engineering
Gothenburg, Sweden
{giaimo, yhang, crnkovic}@chalmers.se, christian.berger@gu.se

ABSTRACT

Establishing and mastering continuous experimentation as an instrument in the portfolio of software product managers is of growing importance resulting in continuous renewal of products for continuous user satisfaction. Product managers for purely software-based products like web-based applications found in online web-shops or smartphone apps can monitor usage profiles of their products in the context of their customers' usage (i.e. the "field"). However, in the area of interconnected embedded systems, cyber-physical systems, or with Internet-of-Things (IoT), such continuous experimentation is under-explored and in many cases understandably not considered due to safety considerations. In this position paper, we are outlining challenges and opportunities of continuous experimentation for cyber-physical systems.

Keywords

Continuous experimentation; cyber-physical systems; safety-critical systems

1. INTRODUCTION

The time in which software is installed on products just once and sold afterwards to customers is over. Today, a growing amount of products like smartphones, watches, fridges, and cars can be continuously monitored and upgraded "in the field". The facilitated access to real world usage profiles allows developers and product analysts to better understand typical usage scenarios in the field, to early recognize products' issues, and to extend their functionalities or fix problems in a much faster way. The continuous experimentation (CE) process is the major accountant for the newly enabled acceleration of products evolution.

The software engineering community has started to realize the importance of this practice; in fact ideas, scope, and principles of CE have been increasingly explored and defined. Due to their context of operations, a growing number of web-based software-intensive companies has already started to apply these ideas in order to ripe the fruits of their technical closeness to end-user data in order to deliver new algorithms and features according to the feedback collected through experiments run on selected sub-sets of their user base.

Most of these studies and applications have however been focusing on the case of pure software-based platforms, but the relationship between CE and Cyber-Physical Systems (CPS) has so far received too little attention. This paper will outline challenges and opportunities realizable by CE for CPS in both non-critical and safety-critical contexts, by providing an overview of the ongoing discussions in the available literature and the state of practice.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

DevOps '16, May 24 - 27, 2016, Edinburgh, UK

© 2016 ACM. ISBN 978-1-4503-2138-9.

DOI: 10.1145/2962695.2962709

2. CONTINUOUS EXPERIMENTATION: STATE OF THE ART

CE is a process that is recently gaining momentum in research and industry. Following the footsteps of Extreme Programming and Agile practices, the idea at its core is to enable product developers to perform controlled experiments in the product's usage context (i.e. post-deployment) and to collect results and statistics thereof. The goal is to use this knowledge for the assessment of issues and the development of new features, resulting in an increased value delivered to the end users over time [2]. In this way, the development process can select focus areas for investing its resources depending on factual evidence collected "in the field".

Based on the evidence gathered in the literature, we can state that while continuous experimentation and its underlying ideas are increasingly understood and accepted, the state of practice is generally not mature: While this key idea is winning over many medium and big sized web-based companies, not all of them operate in a systematic way, due mostly to the organizational constraints and the unsatisfied need to accelerate the development cycle [5].

In general, software companies that have refined their experimentation techniques are medium-to-large web-based corporations with very large user bases, and are able to create a highly automated software deployment and experimentation infrastructure. The purpose of the infrastructure is to build knowledge based on data gathered from the products during their executions, and to enable experimentation processing. In this way, such companies have the chance to perform several hundreds of experiments at the same time, and use statistics to analyze the results obtained from the field of application [4].

However, these pioneer studies and early adopters are limited to the web-based and software-intensive field, and little light has been shed on the application of this process

to the cyber-physical world. In fact, CPS are somehow less favored in this process, due to their proximity to the hardware level and the prolonged development cycle compared to pure software systems. In any case, the underlying principles are the same, and the key to enable the CE process is the post-deployment collection of data. Several case studies have been performed on how some software-intensive embedded systems companies were using post-deployment data, and the results showed that while the process itself provides valuable feedback data [1] as expected, the data collected in the field is still an "untapped resource" for the generality of companies [3]. To the best of the authors' knowledge, no literature was found about CE applied specifically to safety- and mission-critical systems in order to study the impact that such practice could have in this context.

3. CHALLENGES AND OPPORTUNITIES FOR CYBER-PHYSICAL SYSTEMS

Successfully adopting CE in cyber-physical systems is still an under-explored area. We have identified the following five major challenges to be tackled for implementing CE for CPS:

Challenge 1: Hardware constraints. The software of cyber-physical systems runs typically on dedicated hardware, which has a longer development cycle and hence, limiting the degree of software updates. Furthermore, such platforms might also exhibit resource constraints so that a newly released feature with more advanced functionality may dramatically decrease the performances of the system, or even worse, break the system integrity.

Challenge 2: Feedback data from both human customers and other systems. CPS may not only interact with human operators but also communicate with other systems using network infrastructures, e.g., in the context of IoT. Unlike web-based systems, in which CE only focuses on collecting data from the end-user, CPS also raise the need for collecting data from the context of systems-of-systems.

Challenge 3: Safety guarantees. Many CPS are safety-critical, e.g., in automotive, avionic, and medical applications. While a temporary downtime or a slight increase in the responsiveness for web-based systems due to runtime software updates is mostly tolerable, even a transient interruption of service in safety-critical systems could be hazardous and life-threatening.

Challenge 4: Involving more stakeholders. CE has already posed an organizational challenge for web-based software companies by involving further teams in addition to software developers in the R&D process, such as product management and sales teams. To successfully apply CE in CPS complicates even more the organizational matter in the sense that more parties, e.g., hardware engineers, developers, and suppliers, will be further coupled with each other.

Challenge 5: Supportive instruments for CE. While CE for pure software-based systems can be more promptly realized, software updates or extension for CPS need to be prepared well in advance through virtual test environments.

Despite these challenges, we also foresee opportunities that motivate the adoption of CE for the software and system development of CPS. Foremost, the data continuously collected by CE in CPS is a powerful resource for activities such as runtime diagnosis, early problem identification, improvement of system functionality, and delivery of new features and products [6]. Moreover, the rapid change of requirements for today's CPS demands a paradigm shift from conventional plan-driven to agile development practices: In this sense, CE provides great potential to facilitate the software evolution of CPS by frequently and incrementally advancing software quality. Lastly, CE is also very much in line for systems whose functionality is based on learning from environmental or user-provided data, by continuously exploiting data from the field of application also for its development.

4. CONCLUSIONS

In this position paper, we have discussed challenges and opportunities of continuous experimentation (CE) for cyber-physical systems (CPS). While CE is increasingly adopted for pure software-based systems, implementing CE for CPS to properly react on constantly changing market requirements is still an under-explored area. We are currently adopting CE in our laboratory for automated driving, where we envision CIDER: Continuous Integration, Continuous Deployment, and Continuous Experimentation towards Continuous Renewal including functional evolution and innovation. CIDER will combine experimentation in the field with systematic and highly automated virtual testing using simulations and growing amounts of data recordings from CPS in the field.

ACKNOWLEDGEMENTS

This project (COPPLAR) is funded by Vinnova FFI, Diarienr: 2015-04849.

5. REFERENCES

[1] U. Eklund and J. Bosch. Architecture for large-scale innovation experiment systems. In *Software Architecture and European Conference on Software Architecture, 2012 Joint Working IEEE/IFIP Conference on*. IEEE, 2012.

[2] F. Fagerholm, A. S. Guinea, H. Mäenpää, and J. Münch. Building blocks for continuous experimentation. In *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering*. ACM, 2014.

[3] H. Holmström Olsson and J. Bosch. *Lean Enterprise Software and Systems: 4th International Conference, LESS 2013, Galway, Ireland, December 1-4, 2013, Proceedings*, chapter Towards Data-Driven Product Development: A Multiple Case Study on Post-deployment Data Usage in Software-Intensive Embedded Systems. Springer Berlin Heidelberg, 2013.

[4] R. Kohavi, A. Deng, B. Frasca, T. Walker, Y. Xu, and N. Pohlmann. Online controlled experiments at large scale. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2013.

[5] E. Lindgren and J. Münch. *Agile Processes, in Software Engineering, and Extreme Programming: 16th International Conference, XP 2015, Helsinki, Finland, May 25-29, 2015, Proceedings*, chapter Software Development as an Experiment System: A Qualitative Survey on the State of the Practice. Springer International Publishing, 2015.

[6] T. Sauvola, L. E. Lwakatare, T. Karvonen, P. Kuvaja, H. H. Olsson, J. Bosch, and M. Oivo. Towards customer-centric software development: A multiple-case study. In *Software Engineering and Advanced Applications, 2015 41st Euromicro Conference on*, Aug 2015.