Towards a Framework for Continuous Software Engineering

Monalessa Perini Barcellos
Ontology and Conceptual Modeling Research Group (NEMO), Computer Science Department
Federal University of Espírito Santo, Vitória – ES, Brazil
monalessa@inf.ufes.br

ABSTRACT

Characteristics and demands of the modern and digital society have transformed the software development scenario and presented new challenges to software developers and engineers, such as the need for faster deliveries, frequent changes in requirements, lower tolerance to failures and the need to adapt to contemporary business models. The adoption of agile practices has allowed organizations to shorten development cycles and increase customer collaboration. However, this has not been enough. Continuous actions of planning, construction, operation, deployment and evaluation are necessary to produce products that meet customers' needs and behaviors, to make well-informed decisions and identify business opportunities. Thus, organizations should evolve from traditional to continuous and data-driven development in a continuous software engineering approach. Continuous Software Engineering (CSE) consists of a set of practices and tools that support a holistic view of software development with the purpose of making it faster, iterative, integrated, continuous and aligned with business. It is a recent topic of Software Engineering, thus there are many open questions. This paper introduces a CSE framework that represents CSE processes, points out some research questions and discusses proposals to address them.

CCS CONCEPTS

• **Software and its engineering → Software creation and management**

KEYWORDS

Continuous Software Engineering, Framework, Ontology

ACM Reference format:

Monalessa Perini Barcellos. 2020. Towards a Framework for Continuous Software Engineering. In *Proceedings of Brazilian Symposium on Software Engineering. ACM, New York, NY, USA, 6 pages.* https://doi.org/10.1145/3422392.3422469

1 Introduction

Changes and unpredictability in market demands, complex and changing customer needs, and pressure to deliver products faster are challenges faced by many software organizations. Some of the difficulties that need to be overcome when dealing with these challenges are due to the lack of connection between important software development activities such as planning, implementation and deployment [7]. These difficulties are usually accentuated when development adopts a traditional sequential approach, prescribed by the waterfall life cycle model.

To address market demands, many organizations have adopted agile methods with the intention of increasing the organization responsiveness to change. By emphasizing flexibility, efficiency and speed of delivery, agile practices have led to a paradigm shift in how software is developed [8][26]. In the last two decades, it has been perceived that increasing the frequency of some critical development activities contributes to reduce some problems. Practices such as *"release early, release often"* have been established mainly in the context of open source software development and have proven beneficial in terms of quality and consistency [24]. The successful adoption of agile methods has also provided evidence of the need for greater flexibility and adaptation in the software development environment [27].

Some initiatives have emerged aiming to speed up the development process and improve the connection between its activities. For example, Continuous Integration [3] seeks to eliminate discontinuities between development and delivery. In a similar approach, DevOps [6] recognizes that the integration between software development and software operation must be continuous. Extending the need for integration to other levels, BizDev [7] advocates that continuity should exist not only in the software process context, but also between software and strategic processes of the organization.

The need for continuous software planning, building, operation and evaluation is addressed in Continuous Software Engineering (CSE), which understands that the software development process is not a sequence of discrete activities, performed by distinct and disconnected teams. It aims to establish a continuous flow between software-related activities, taking into consideration the entire software life cycle. It is a recent topic that seeks to transform discrete development practices into more iterative, flexible and continuous alternatives, keeping the goal of building and delivering quality products according to established time and costs [7].

As a recent topic, there are several emergent works (e.g., [6][7][18][19][26]) but there are also many issues to be

SBES '20, October 21–23, 2020, Natal, Brazil

ACM ISBN 978-1-4503-8753-8/20/09…$15.00
https://doi.org/10.1145/3422392.3422469

addressed. For example, a formal description of CSE aspects and involved processes is needed to enable its adoption in real world settings [12]. Moreover, it is necessary to provide knowledge to overcome challenges in the introduction and enhancement of CSE in companies [21][17]. In fact, a general conceptualization to provide understanding about CSE as a whole is needed. Thus, in this paper, we introduce a framework for CSE, representing a set of processes to be performed in the CSE context and the main relations between them. We also discuss some research opportunities and introduce some proposals to address them. The paper is organized as follows. Section 2 provides the background for the paper. Section 3 introduces the proposed CSE framework. Section 4 discusses research questions and proposals of solution. Section 5 concludes the paper.

2 Background

CSE involves practices and tools that aims at establishing an end-to-end flow between customer demand and the fast delivery of a product or service. The 'big picture' by which this might be achieved goes beyond agile principles and surfaces a more holistic set of continuous activities [7]. According to Johanssen et al. [17], in CSE, customers are proactive, and users and other stakeholders are involved in the process, learning from usage data and feedback. Planning is continuous, so as requirements engineering, which focuses on features, modularized architecture and design, and fast realization of changes. Agile practices are employed, including short development cycles, continuous integration of work, continuous delivery and continuous deployment of releases. There is version control of code, branching strategies, fast commit of code, code coverage and code reviews. Quality assurance involves automated tests, regular builds, pull requests, audits and run-time adaption. Knowledge is shared and continuous learning happens, capturing decisions and rationale.

In the last years, some works have addressed CSE processes and practices. Olsson et al. [26] defined the Stairway to Heaven model (StH), which describes the evolution path organizations follow to successfully move from traditional to customer data-driven software development. It comprises five stages: traditional development, agile organization, continuous integration, continuous deployment, and R&D as an innovation system. In summary, organizations evolving from traditional development start by experimenting with one or a few agile teams. Once these teams are successful, agile practices are adopted by the organization. As the organization starts showing the benefits of working agile, system integration and verification becomes involved and the organization adopts continuous integration. Once continuous integration runs internally, lead customers often express an interest to receive software functionality earlier than through the normal release cycle. They want continuous deployment of software. The final stage is where the organization collects data from its customers and uses the installed customer base to run frequent feature experiments to support customer data-driven software development.

Fitzgerald and Stol [7] argue that continuous activities go beyond software engineering activities. They introduce the Continuous* term, as a set of activities from business, development, operations and innovation that provides the holistic view of the software life cycle. Continuous planning, continuous security, continuous use, continuous trust and continuous experimentation are some of the Continuous* activities considered by the authors. They introduce BizDev, analogous to DevOps, but referring to the continuity and alignment between business strategy and software development.

From interviews performed with CSE practitioners, Johanssen et al. [16] defined the Eye of CSE, which consists of 33 elements (e.g., practices) organized in nine categories. According to the authors, the Eye of CSE can serve as a checklist for practitioners to tackle the subject of CSE by incrementally applying CSE elements and keeping an eye on potential next steps. The proposal differs from the sequential nature of the StH model [26]. Even if some CSE elements, such as continuous integration and delivery, require a stepwise introduction, the authors argue that CSE should be approached from multiple angles simultaneously.

Some studies have investigated specific aspects related to CSE, such as user feedback [15], practical issues [16], human factors [28] and decision knowledge [20]. Some of them reveal challenges and difficulties faced in the CSE context. From works reported in the literature, it can be noted that a consensus about what CSE is and its practices has not been achieved yet [16].

3 A Framework for CSE

Aiming to provide knowledge about CSE, we have worked on a framework that represents a set of processes to be performed in the CSE context and the main relations between them. Its purpose is to provide an overview of CSE and serve as a basis for future research. It was inspired mainly by the works by Olsson et al. [26], Fitzgerald and Stol [7] and Johanssen et al. [16]. The processes were defined taking processes suggested in [7] and stages defined in [26] into account. Different from [26], our framework considers that processes can be performed simultaneously and gradually. Elements from [16] were also considered to define the processes (e.g., Continuous Knowledge Management was based on knowledge-related elements of the Eye of CSE). Figure 1 shows an overview of the proposed framework. In the figure, processes containing development activities and that are at the core of CSE are represented by pentagons, rectangles with rounded edges represent other processes, thin arrows represent data flow from/to the repository and wide arrows represent information flow between process. Data flows mean that the process produces and stores data in the repository, or uses data stored in it (for example, Agile Development stores in the repository data about effort spent on the tasks). Information flows mean that processes change information, which does not necessarily involve data from the repository (for example, Continuous Quality Assurance can establish new quality standards/goals to be used in Agile

Development). For visualization reasons, information flows between core processes are omitted in the figure (they are implicit in the shape used to represent these processes). Moreover, Continuous Knowledge Management and Continuous Software Measurement are represented as vertical rectangles, meaning that they are related to all the other processes.

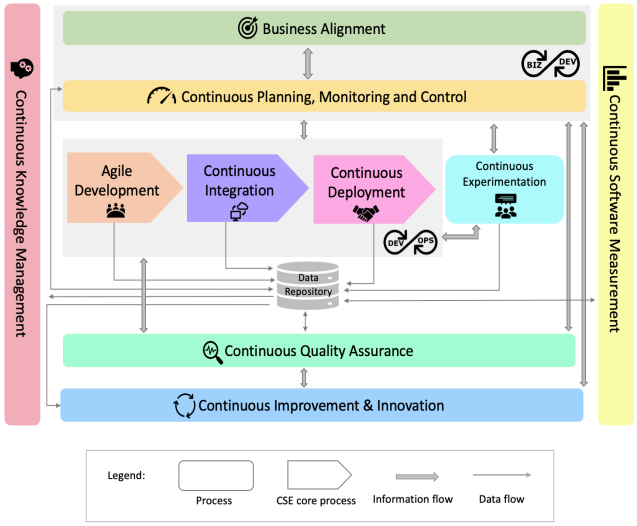

Figure 1: Framework for CSE - Overview

In a nutshell, organizations adopt **Agile Development** practices, which include, among others, the notion of small and empowered teams, backlog, short time-box and daily standup meetings. Once agile development has been adopted, there is the need to test the built code in the broader context of the system. This leads to the adoption of **Continuous Integration**, which involves, among others, test-driven development, automated build and test environment. When the customer demands more frequent releases of software, the organization performs **Continuous Deployment,** where software is deployed at customers at the end of agile sprints, or even more frequently, and after passing the continuous integration testing activities. These three processes working together embody the DevOps approach [6]. Organizations can perform **Continuous Experimentation**, which involves experiments (e.g., tests A/B) to continuous evaluate new features and optimize existing features by considering customer data and feedback. Over the processes, quality assurance activities (e.g., verification and validation) are continuously performed (**Continuous Quality Assurance**) aiming at product and process quality.

Continuous Software Measurement define measures necessary to provide useful information for daily activities and decision making (e.g., number of sprints, planned tasks, performed tasks, team productivity, number of defects, number of deployments, user feedback, etc.). During the execution of the processes, data is collected and stored in the data repository. Data is used to aid in **Continuous Planning, Monitoring and Control** and support data-driven decision making. Based on information obtained from data analysis, plans are reviewed, new plans are established and corrective actions are performed in alignment to the business goals (**Business Alignment**), which is consistent with the BizDev approach [7]. Data from the repository combined to management and business information are used to **Continuous Improvement and Innovation** of products and processes. Data provided from users in the experiments are particularly important in this context. Finally, **Continuous Knowledge Management** disseminates knowledge useful to perform the processes and also captures, evaluates and makes available new knowledge produced when processes are performed.

Concerning the relations among the processes, Business Alignment provides business information to Continuous Planning, Monitoring and Control that, in turn, provides management information regarding software projects, aiming to align business and software projects. These processes exchange information with the core processes to carry out development activities according to business and projects goals. The relations between these processes allow to connect and align aspects from BizDev and DevOps approaches.

Continuous Experimentation uses information from the core processes to evaluate features and provides information to these processes to improve existing features or develop new ones. Information from the experiments results are used in Continuous Planning, Monitoring and Control to adjust or create plans according to the features to be developed or adjusted. They are also important to reveal new opportunities in Business Alignment.

Continuous Quality Assurance uses information from the core processes (e.g., requirements, produced artifacts, adopted methods) and inform them about quality standards/goals to be met and quality assurance results. This information is also relevant to adjust plans considering the actions necessary to correct non-conformities. It is also useful to Continuous Improvement and Innovation, helping identify improvements that can be made in software products and processes. Information about improvements and innovation is relevant to the establishment of new quality standards and goals. Continuous Improvement and Innovation needs to take management and business information into account. On the other side, improvements and innovation actions need to be aligned to business and considered in new plans.

Continuous Software Measurement identifies information needs from all the processes and define measures to be collected to meet the information needs. Lastly, Continuous Knowledge Management captures knowledge from all the process and also provides knowledge to all of them.

4. Research Questions and Proposals of Solution

We have used our framework as starting point to identify research questions that can be explored in future research. Our plan is to advance on the research and add the produced results to the framework, turning it into a more robust tool to support

CSE. In this section, we discuss some questions and some proposals on which we have worked to address the questions.

RQ1. How to implement CSE practices and evolve from a practice to another?

Implement CSE practices requires changes in the way the organization works. Many times, organizations struggle with the changes to be made along the path and with the order in which to implement them [26]. There are some proposals that aim to help organizations in this matter, such as [26] and [19]. However, it is necessary to grow knowledge (best practices, guidelines, models, approaches, etc.) about how CSE processes can be implemented. Moreover, different organizational contexts need to be taken into account (e.g., startup, software factory, product-based companies) because CSE practices must be tailored to fit the business goals, culture, environment and other aspects of the organization. In this sense, we have explored the use of Systems Thinking [23] theory and tools to support understanding the organization and, thus, define actions to implement CSE practices according to the organization characteristics, problems and priorities. We have proposed and experienced a systems thinking-based process to support organizations to define actions to implement CSE practices [31]. From that and future experiences and studies, we intend to define a set of guidelines to aid organizations to identify how to implement CSE practices according to their characteristics.

RQ2. In CSE, which are the involved processes/activities, resources, artifacts and stakeholders? For example, what does constitute continuous integration?

CSE involves several processes, including development, management, supporting and business processes, among others. In the context of Software Engineering as a whole, there are standards (e.g., [14]), maturity models (e.g., [33]) and a vast literature defining and detailing processes. However, as a recent topic, there is a lack of such definitions in the CSE context. Without a clear definition about the processes involved in CSE, researchers and practitioners may have different understandings about CSE. For example, practitioners often define CSE driven by processes closer to development [16], while researchers tend to see the big picture [7]. Our framework and works such as [7] help identify processes involved in CSE. However, it is necessary to detail the processes, its elements (e.g., inputs, outputs, roles), and clearly represent the relations between them. Moreover, it is necessary to explain how they differ from 'traditional' processes.

Processes can be defined by means of textual descriptions or by using conceptual modeling principles and tools, such as in [21]. Ontologies (more specifically reference ontologies) can be particularly useful in this matter. A reference ontology is a special kind of conceptual model representing a model of consensus within a community. It is a solution-independent specification with the aim of making a clear and precise description of the domain of interest in reality for the purposes of communication, learning and problem-solving [11]. Ontologies can describe a particular domain (so called domain ontologies) or a task/process (so called task ontologies) [10]. They have been successfully used to represent knowledge and solve knowledge-related problems in Software Engineering (e.g., [2][1][34]). Therefore, as we are interested in detailing CSE processes, we have proposed the use of task ontologies to provide knowledge about CSE processes. A task ontology clearly represents the process and provides knowledge that enables to answer the following general questions regarding the process being addressed: (i) which are the process activities? (ii) Who is responsible for performing them? (iii) How the activities are decomposed into sub-activities? (iv) What is the control flow between them? (v) What are the inputs and outputs of each activity?. Currently, we are working on task ontologies referring to the Continuous Integration and Continuous Deployment processes of our framework.

RQ3. Which tools can be used to support the processes?

Automated tools are crucial to implement CSE. There is a large set of tools available for organizations to use to support CSE processes [32]. The tools used by an organization directly influence the way it performs CSE. There are cases in which the adopted tools keep the organization from making a complete transition and fully adapting CSE. Practitioners, in particular developers and CSE specialists, often rely on a tool-driven approach for defining CSE. Commercially available tools influence their understanding of CSE [16]. The selection of the tools to compose the 'CSE tool chain' must be careful and take technological and organizational aspects into account. In this sense, we have investigated criteria to select tools to support CSE processes (e.g., required technology to use the tool, tool price, potential to integration to others tools used by the organization, team experience with the tool, supported processes/activities, ease of use, user support, etc.). We intend to reach a set of criteria suitable for CSE context and that can be customized according to the organization context and interests. For example, depending on the organization characteristics, some criteria can be disregarded (e.g., if the organization requires the use of free tools, tool price criterion is not considered). Moreover, criteria can have different weights for different organizations.

RQ4. How to obtain integrated data to support software development and decision making in CSE?

During the execution of CSE processes, data is produced and stored in different ways. For example, the tools used to support the processes collect and store data regarding the supported processes. Electronic spreadsheets are also commonly used to store complementary management data. According to Svensson et al. [35], despite the vast amount of data stored in tools, decisions on software development are commonly based on subjective aspects, such as previous experiences of the managers and stakeholders, intuitions or a combination of these. One of the reasons organizations fail to leverage data stored in tools is the difficulty to access, integrate, analyze and view data handled by heterogeneous tools. Due to its continuous nature, in CSE the need for integrated data increases. CSE values data-driven software development [7][26], in which data about the processes,

team, products, clients and organization are used to support daily activities and decision-making. Therefore, it is necessary to integrate data produced by different agents and tools and provide integrated information to support well-informed decisions. However, integration is not an easy task. One source of difficulty for data integration is semantic heterogeneity, which can result in conflicts whenever the same information item is given divergent interpretations, a situation which may not even be detected [36]. Neglecting these semantic conflicts can lead to incorrect integration and wrong information for decision-making. To reduce these conflicts, integration should address semantic issues. Ontologies have become the predominant way to deal with semantics in semantic integration initiatives [25]. They can be used to establish a common conceptualization about the domain in order to support communication and data integration. They work as an interlingua to map the concepts used by different tools and data sources, enabling data and services understanding [5]. Inspired by works such as [5], [9] and [29], we have developed and used domain ontologies as reference models to integrate tools that support CSE processes. We started by developing a Scrum Reference Ontology and integrating it to ontologies from the Software Engineering Ontology Network (SEON) [30], covering requirements and project management aspects. We applied the ontology as a basis to integrate tools (e.g., Clockify and Azure DevOps) used by development teams in the software unit of a Brazilian government agency. As a result, data from different tools were integrated and shown in dashboards, providing useful information for managers to make decisions.

RQ5. Which measures can be used in CSE?

An understanding of the organization capabilities to achieve business goals can only be obtained through measuring [13]. This varies from business measures (e.g., revenue) to management (e.g., team productivity) and development-related measures (e.g., test coverage, time to deploy a new release). Therefore, it is necessary to define appropriate measures to properly evaluate the use of CSE practices in the organization. Measures are also particularly important to continuous experimentation, since they quantify experiments results and facilitate measuring the value-add of specific product features. Identifying relevant measures is of high importance in order to collect data that will work as a basis for product improvement and development of new features [4]. We have started to investigate measures to support data-driven software development and decision making in CSE. RQ5 is strongly connected to RQ4 because data provided from integrated tools should be related to measures defined to meet the organization information needs. Therefore, we have made efforts to develop a semantic integration platform to serve as infrastructure to support measurement and data integration. The platform uses ontologies from SEON [30] (and new ontologies we have developed to address CSE aspects), to build semantic services that aid in tools integration to enable extraction of data related to the measures that meet the organization information needs.

RQ6. How to collect feedback from users? How to use users' feedback to support process and product improvement and identify new business opportunities?

In CSE, new versions of the software product are delivered to the client more often. This enables developers to frequently retrieve user feedback on the latest software increment. Moreover, in a CSE environment it is possible to carry out experiments to evaluate features to be incorporated to the product. However, there is still no well-established processes to interact with users in a CSE environment [16] (this issue is also related to RQ2). User feedback can be collected in an explicit manner (e.g., through forms and written reviews), but monitoring the implicit user feedback is also important to improve understanding the need for new requirements [22]. Moreover, the organization needs to develop the capability to effectively use collected data about user feedback to test new ideas with customers [4]. Aiming to address this research question, we will associate results related to RQ2, RQ4 and RQ5. We intend to provide knowledge about the processes that collect feedback from users (RQ2), providing an integrated view of user feedback in a CSE environment. We also plan to investigate methods and techniques that can be adopted to support obtaining user feedback and stimulating user to make explicit his/her implicit impressions. We will extend our data integration and measurement solution (RQ4 and RQ5) to cover user feedback data considering measures that meet information needs. By analyzing data, it will be possible to verify alignment between business goals and software development, improve products and identify new opportunities (e.g., new requirements that can be incorporated to the product, or even a new product).

5 Final Considerations

This paper introduced a framework representing processes involved in a CSE environment. As we said, the processes constituting our framework were selected considering mainly [7], [16] and [26]. However, our work differs from these. Fitzgerald and Stol [7] define a set of continuous activities and organize them into categories. The authors do not discuss how the activities relate to each other in a CSE environment. Olsson et al. [26] define sequential stages to implement CSE practices, but some aspects, such as quality assurance and knowledge management, are not considered. Johanssen et al. [16] identify some CSE practices without relating them to processes. Different from these works, our framework identifies CSE processes and the main relations among them by means of information and data flows. The framework provides a more comprehensive view of CSE than [26] and details aspects not covered in [7]. By providing an overview of a CSE environment, the framework can help practitioners to better make decisions about how to implement it. The framework aids to implement CSE practices considering the big picture, instead of each process in isolation. For researchers, it can serve as a starting point to future research. In this sense, we have identified some research questions and worked on solutions to address them.

We plan to use results from our works related to the RQs to improve the framework, making it more robust. For example, we intend to: detail the framework processes by defining their subprocesses, activities, artifacts and involved roles (RQ2); provide a catalog of tools to support the processes and a set of criteria to select the ones more suitable for an organization (RQ3); make available an infrastructure to the data repository and a semantic integration approach to integrate tools aiming to produce integrated data (RQ4); provide a catalog of measures related to the processes (RQ5); add a process to support organization analysis to define strategies to implement CSE (RQ1). The framework can also be extended to include other processes (e.g., processes related to human resources). Moreover, methods and practices such as the ones cited in [16] can be mapped to the processes to increase knowledge for their execution. Furthermore, other solutions can be proposed to address the research questions

In this paper, we explored six research questions. Other questions can be investigated. For example, there are many issues related to knowledge management. On one side, CSE is strongly based on knowledge. On the other side, agile practices tend to neglect knowledge recording and storage. So, how to deal with that in a balanced way? The work described in this paper is a work in progress. From it, we expect to enable advances in CSE state of the art and state of the practice as well as motivate other researchers to do so.

REFERENCES

[1] Monalessa P. Barcellos and Ricardo A. Falbo. 2013. A software measurement task ontology. In *Proceedings of the 28th ACM Symposium on Applied Computing*, 311–318. https://doi.org/10.1145/2480362.2480428

[2] Monalessa P. Barcellos, Ricardo A. Falbo, and Ana Regina Rocha. 2013. A strategy for preparing software organizations for statistical process control. *Journal of the Brazilian Computer Society* 19, 4.

[3] Kent Beck. 2000. *Extreme Programming Explained: Embrace Change.* Addison-Wesley.

[4] Jan Bosch (ed.). 2014. *Continuous Software Engineering*. Springer.

[5] Rodrigo F. Calhau and Ricardo A. Falbo. 2010. An Ontology-based Approach for Semantic Integration. In *Proceedings 14th IEEE International Enterprise Distributed Object Computing Conference*, 111–120.

[6] Patrick Debois. 2011. Devops: a software revolution in the making? *Cutter IT Journal* 24, 8.

[7] Brian Fitzgerald and Klaas-Jan Stol. 2017. Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software* 123: 176–189.

[8] Nina D. Fogelström, Tony Gorschek, Mikael Svahnberg, and Peo Olsson. 2010. The impact of agile principles on market-driven software product development. *Journal of Software Maintenance and Evolution: Research and Practice* 22, 1: 53–80. https://doi.org/10.1002/spip.420

[9] Vinícius .S. Fonseca, Monalessa P. Barcellos, and Ricardo A. Falbo. 2017. An ontology-based approach for integrating tools supporting the software measurement process. *Science of Computer Programming* 135: 20–44. https://doi.org/10.1016/j.scico.2016.10.004

[10] Nicola Guarino. 1998. Formal Ontology and Information Systems. *In: Proceedings of the International Conference in Formal Ontology and Information Systems - FOIS'98, Trento, Italy*: 3–15.

[11] Giancarlo Guizzardi. 2007. On Ontology, Ontologies, Conceptualizations, Modeling Languages and (Meta)Models. *Frontiers in Artificial Intelligence and Applications, Databases and Information Systems IV. IOS Press, Amsterdam.*

[12] Jez Humble and David Farley. 2010. *Continuous delivery: reliable software releases through build, test, and deployment automation.* Pearson.

[13] Jez Humble and Joanne Molesky. 2011. Why enterprises must adopt devops to enable continuous delivery. *CUTTER IT JOURNAL* 24, 8.

[14] ISO/IEC. 2008. ISO/IEC 12207:2008 - Systems and Software Engineering - Software Life Cycle Process . *International Organization for Standardization and the International Electrotechnical Commission, Geneva, Switzerland.*

[15] Jan O. Johanssen, Anja Kleebaum, Bernd Bruegge, and Barbara Paech. 2019. How do Practitioners Capture and Utilize User Feedback During Continuous Software Engineering? In *IEEE 27th International Requirements Engineering Conference (RE)*, 153–164. https://doi.org/10.1109/RE.2019.00026

[16] Jan O. Johanssen, Anja Kleebaum, Barbara Paech, and Bernd Bruegge. 2018. Practitioners' Eye on Continuous Software Engineering: An Interview Study. In *Proceedings of the International Conference on Software and System Process*, 41–50. https://doi.org/10.1145/3202710.3203150

[17] Jan Ole Johanssen, Anja Kleebaum, Barbara Paech, and Bernd Bruegge. 2019. Continuous software engineering and its support by usage and decision knowledge: An interview study with practitioners. *Journal of Software: Evolution and Process* 31, 5: e2169. https://doi.org/10.1002/smr.2169

[18] Teemu Karvonen, Lucy E.T. Lwakatare, Tanja Sauvola, Jan Bosch, Helena H. Olsson, Pasi Kuvaja, and Markku Oivo. 2015. Hitting the Target: Practices for Moving Toward Innovation Experiment Systems. In *International Conference of Software Business*, 117–131.

[19] Teemu Karvonen, Tanja Suomalainen, Marko Juntunen, Tanja Sauvola, Pasi Kuvaja, and Markku Oivo. 2016. The CRUSOE Framework: A Holistic Approach to Analysing Prerequisites for Continuous Software Engineering. In *17th International Conference on Product-Focused Software Process Improvement*, 643–661.

[20] Anja Kleebaum, Jan O. Johanssen, Barbara Paech, Rana Alkadhi, and Bernd Bruegge. 2018. Decision Knowledge Triggers in Continuous Software Engineering. In *Proceedings of the 4th International Workshop on Rapid Continuous Software Engineering*, 23–26. https://doi.org/10.1145/3194760.3194765

[21] Stephan Krusche and Bernd Bruegge. 2017. CSEPM - A Continuous Software Engineering Process Metamodel. In *IEEE/ACM 3rd International Workshop on Rapid Continuous Software Engineering*, 2–8. https://doi.org/10.1109/RCoSE.2017.6

[22] Walid Maalej, Hans-Jörg Happel, and Asarnusch Rashid. 2009. When Users Become Collaborators: Towards Continuous and Context-Aware User Input. In *Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications*, 981–990. https://doi.org/10.1145/1639950.1640068

[23] Donella Meadows. 2008. *Thinking in Systems: A Primer.* Chelsea Green Publishing Company.

[24] Martin Michlmayr, Brian Fitzgerald, and Klaas-Jan Stol. 2015. Why and How Should Open Source Projects Adopt Time-Based Releases? *IEEE Software* 32, 2: 55–63. https://doi.org/10.1109/MS.2015.55

[25] Julio C. Nardi, Ricardo A. Falbo, and João Paulo A. Almeida. 2013. Foundational Ontologies for Semantic Integration in EAI: A Systematic Literature Review. In *Proceedings 12th IFIP WG 6.11 Conference on e-Business, e-Services, and e-Society, I3E 2013*, 238–249.

[26] Helena H. Olsson, Hiva Alahyari, and Jan Bosch. 2012. Climbing the "Stairway to Heaven" - A Mulitiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software. In *2012 38th Euromicro Conference on Software Engineering and Advanced Applications*, 392–399. https://doi.org/10.1109/SEAA.2012.54

[27] Efi Papatheocharous and Andreas S Andreou. 2014. Empirical evidence and state of practice of software agile teams. *Journal of Software: Evolution and Process* 26, 9: 855–866. https://doi.org/10.1002/smr.1664

[28] Efi Papatheocharous, Marios Belk, Jaana Nyfjord, Panagiotis Germanakos, and George Samaras. 2014. Personalised Continuous Software Engineering. In *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering*, 57–62. https://doi.org/10.1145/2593812.2593815

[29] Laylla D. C. Renault, Monalessa P. Barcellos, and Ricardo A. Falbo. 2018. Using an Ontology-based Approach for Integrating Applications to Support Software Processes. In *Proc. of the XVII Brazilian Symposium on Software Quality.*

[30] Fabiano Ruy, Ricardo A. Falbo, Monalessa P. Barcellos, Simone D. Costa, and Giancarlo Guizzardi. 2016. SEON: A software engineering ontology network. In *Proceedings of the 20th International Conference on Knowledge Engineering and Knowledge Management.* https://doi.org/10.1007/978-3-319-49004-5_34

[31] Paulo Sérgio Santos Jr, Monalessa P. Barcellos, and Rodrigo F. Calhau. 2020. Am I going to Heaven? First step climbing the Stairway to Heaven Model - Results from a Case Study in Industry. In *Proceedings of the 34th Brazilian Symposium on Software Engineering (SBES).*

[32] Mojtaba Shahin, Muhammad A. Babar, and Liming Zhu. 2017. Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices. *IEEE Access* 5: 3909–3943.

[33] Software Engineering Institute. 2018. *Capability Maturity Model Integration (CMMI 2.0).*

[34] Érica F. Souza, Ricardo A. Falbo, and Nandamudi L. Vijaykumar. 2017. ROoST: Reference Ontology on Software Testing. *Applied Ontology* 12: 59–90.

[35] Richard B. Svensson, Robert Feldt, and Richard Torkar. 2019. The Unfulfilled Potential of Data-Driven Decision Making in Agile Software Development. In *Agile Processes in Software Engineering and Extreme Programming*, 69–85.

[36] Holger Wache, Thomas Vogele, Ubbo Visser, Heiner Stuckenschmidt, Gerhard Schuster, H. Neumann, and S. Hubner. 2001. Ontology-Based Information Integration: A Survey of Existing Approaches. In *Proceedings of the IJCAI'01 - Workshop: Ontology and Information Sharing*, 108–117.