


Continuous Software Engineering

Monalessa Perini Barcellos

monalessa@inf.ufes.br

Ontology and Conceptual Modeling Research Group (NEMO)
Computer Science Department
Federal University of Espírito Santo (UFES)


1

Introduction

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 understands that the **software development process is not a sequence of discrete activities**, performed by distinct and disconnected teams or departments.

(Fitzgerald and Stol, 2017)



2

Introduction

The objective of Continuous Software Engineering is **to establish a continuous flow between all software-related activities, taking into consideration their entire life cycle.**

It is a recent subarea of Software Engineering that **seeks to transform discrete development practices into more iterative, flexible and continuous alternatives**, maintaining the objective of building and delivering quality products according to established deadlines and costs.

(Fitzgerald and Stol, 2017)

It is a recent area, thus there are many open questions.



3

Some Practices related to CSE

Lean Thinking: continuous movement, flow between activities/people, automation, continuous improvement, business alignment.

*Agile x Lean: whereas **agile** software development is mostly focused on the software development function, **Lean Thinking** has a very explicit focus on the end-to-end process: from customer to delivery.*

Enterprise Agile (scaling up Agile): “the ability of organizations to sense **environmental change** and respond appropriately” (e.g., SAFe).

DevOps: principles (**culture, automation, measurement, sharing**)

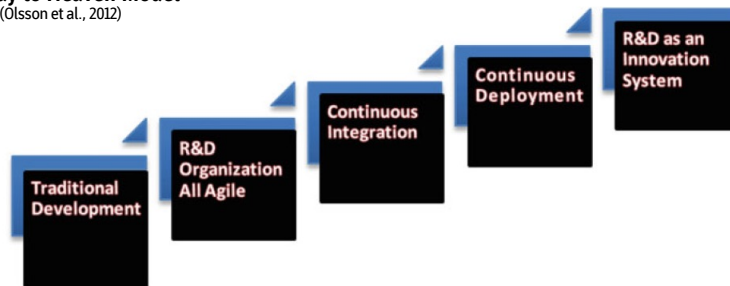
(Fitzgerald and Stol, 2017)



4

Some CSE Proposals

Stairway to Heaven Model (Olsson et al., 2012)



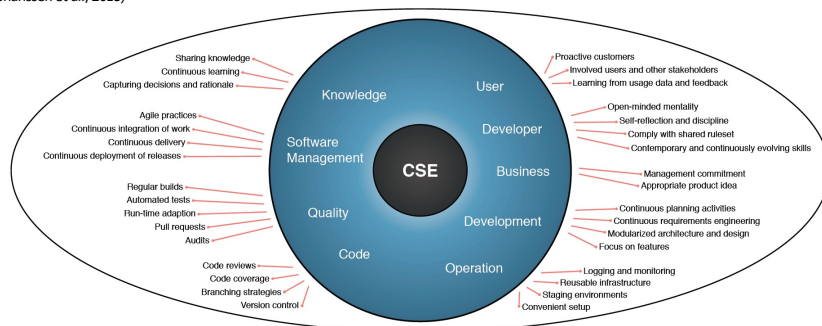
It defines sequential stages to implement CSE practices.

nemo

5

Some CSE Proposals

Eye of CSE (Johansen et al., 2018)



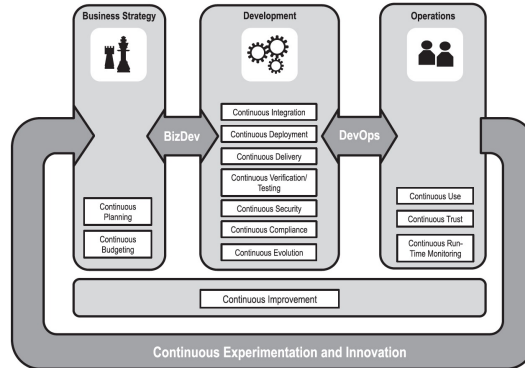
Identifies and categorizes some CSE practices.

nemo

6

Some CSE Proposals

Continuous*
(Fitzgerald and Stol, 2017)



It defines a set of continuous activities and organize them into categories.



Some CSE Proposals

Continuous*
(Fitzgerald and Stol, 2017)

Continuous* activities and definitions.

Activity Description and references

Business strategy and planning

Continuous planning Holistic endeavor involving multiple stakeholders from business and software functions whereby plans are dynamic open-ended artifacts that evolve in response to changes in the business environment, and thus involve a tighter integration between planning and execution. (See Knight et al. (2001), Myers (1990), Lehtisaari et al. (2008)).

Continuous budgeting Budgeting is traditionally an annual event, during which an organization's investments, revenue and expense outlook are prepared for the year to come. The Beyond Budgeting model suggests that budgeting becomes a continuous activity, also, to better facilitate changes during the year. (See Bigoski (2008), Frow et al. (2010), Lohani (2013)).

Development

Continuous integration A typically automatically triggered process comprising inter-connected steps such as compiling code, running unit and acceptance tests, validating code coverage, checking coding standard compliance and building deployment packages. While some form of automation is typical, the frequency is also important in that it should be regular enough to ensure quick feedback to developers. Finally, any continuous integration failure is also an important event which may have a number of ceremonies and highly visible artifacts to help ensure that problems leading to integration failures are solved as quickly as possible by those responsible. (See Kim et al. (2008), Lacoste (2009), Rogers (2004), Stahl and Bosch (2013), Stolberg (2009)).

Continuous delivery Continuous delivery is the practice of continuously deploying good software builds automatically to some environment, but not necessarily to actual users (See Neely and Stolt (2013), Humble and Farley (2010)).

Continuous deployment Continuous deployment implies continuous delivery and is the practice of ensuring that the software is continuously ready for release and deployed to actual customers (see Claps et al. (2015), Fitz (2009), Holmström Olsson et al. (2012)).

Continuous verification Adoption of verification activities including formal methods and inspections throughout the development process rather than relying on a testing phase towards the end of development. (See Chang et al. (1997), Cordoni et al. (2016)).

Continuous testing A process typically involving some automation of the testing process, or prioritization of test cases, to help reduce the time between the introduction of errors and their detection, with the aim of eliminating root causes more effectively. (See Borsari et al. (2012), Marjan et al. (2013), Muslu et al. (2013), Saff and Ernst (2003)).

Continuous compliance Software development seeks to satisfy regulatory compliance standards on a continuous basis, rather than operating a "big bang" approach to ensuring compliance just prior to release of the overall product. (See Fitzgerald et al. (2013), McHugh et al. (2013)).

Continuous security Transforming security from being invoked as just another non-functional requirement to a key concern throughout all phases of the development lifecycle and even post deployment, supported by a smart and lightweight approach to identifying security vulnerabilities. (See Merkow and Raghavan (2011)).

Continuous evolution Most software systems evolve during their lifetime. However, a system's architecture is based on a set of initial design decisions that were made during the system's creation. Some of the assumptions underpinning these decisions may no longer hold, and the architecture may not facilitate certain changes. In the last years, there has been increased focus on this topic. When an architecture is unsuitable to facilitate new requirements but shortcuts are made nevertheless, technical debt is incurred. (See Dell Rosso (2008), Riaz et al. (2009)).

Operations

Continuous use Recognizes that the initial adoption versus continuous use of software decisions are based on different parameters, and that customer retention can be a more effective strategy than trying to attract new customers. (See Bhattacharjee (2001), Gohauer et al. (2013), Ortiz de Guzman and Markus (2009)).

Continuous trust Trust developed over time as a result of interactions based on the belief that a vendor will act cooperatively to fulfill customer expectations without exploiting their vulnerabilities. (See Gefen et al. (2003), Hoehle et al. (2012), Zhou (2013)).

Continuous run-time monitoring As the historical boundary between design-time and run-time research in software engineering is blurring (Sarani and Chenni, 2010), in the context of continuously running cloud services, run-time behaviors of all blocks must be monitored to enable early detection of quality-of-service problems, such as performance degradation, and also the fulfillment of service level agreements (SLAs). (See van Hooten et al. (2009)).

Improvement and Innovation

Continuous improvement Based on lean principles of data-driven decision-making and elimination of waste, which lead to small incremental quality improvements that can have dramatic benefits and are hard for competitors to emulate. (See Chen et al. (2007), Fowler (1999), Järvinen et al. (1999), Krasner (2002)).

Continuous innovation A sustainable process that is responsive to evolving market conditions and based on appropriate metrics across the entire lifecycle of planning, development and run-time operations. (See Cole (2012), Holmström Olsson et al. (2012), Riaz (2013)).

Continuous experimentation A software development approach based on experiments with stakeholders consisting of repeated Build-Measure-Learn cycles. (See Adams et al. (2013), Bosch (2012), Eggertson et al. (2014), Steiber and Alänge (2013)).



CSE Challenges

Win the war, not the battles: Focus on the holistic view of the end-to-end process instead of getting worried about each technique to be implemented (some of them may not be needed in an organization)

Context and Culture: There are numerous dimensions in which contexts vary, for instance the business domain in which organizations operate. Context and culture must be taken into account when defining which continuous process apply to an organization.

Misplaced focus on speed rather than continuity. "speed is meaningless without continuity". For software engineering, achieving flow and continuity is much more important in first instance than speed.

The need for discontinuous software engineering. creativity and innovation require discontinuous thinking—in some cases an abrupt, discontinuous change is needed rather than a gradual change.

BizDev. there are fundamental mismatches between expectations and planning strategies of business and sales managers on the one hand, and the capability to deliver functionality in a timely fashion on the other hand.

Measurement. Part of realizing the Continuous* involves the development of appropriate metrics that can help in predicting how fast interactions with external communities and other players in an ecosystem can be achieved. Metrics will also be important in realizing the continuous experimentation concept, including the suggested 'feature analytics,' which will facilitate measuring the value-add of specific features .

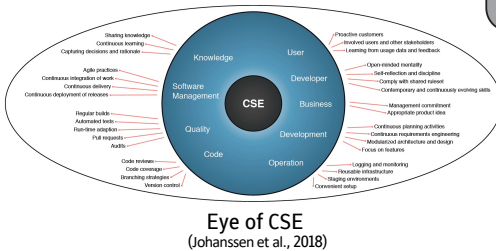
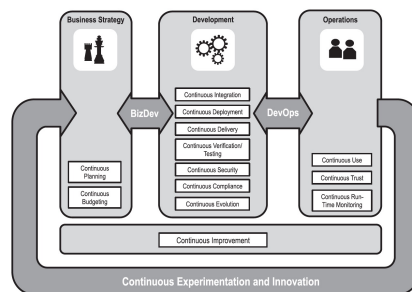
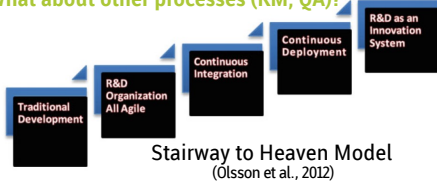


9

Some issues remain...

What to do at each stage?

What about other processes (KM, QA)?

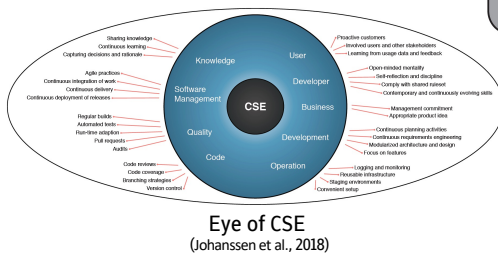
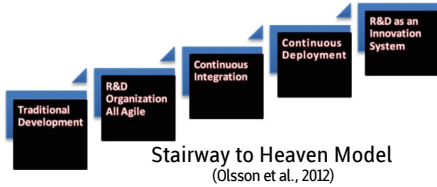


10

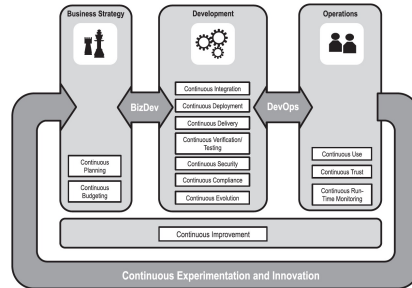
Some issues remain...

What to do at each stage?

What about other processes (KM, QA)?



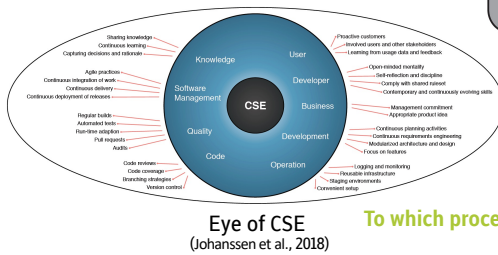
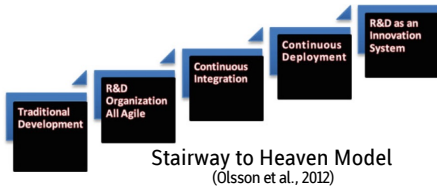
How do the activities/processes relate to each other in a CSE environment?



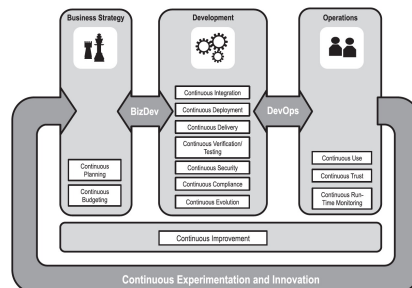
Some issues remain...

What to do at each stage?

What about other processes (KM, QA)?



How do the activities/processes relate to each other in a CSE environment?

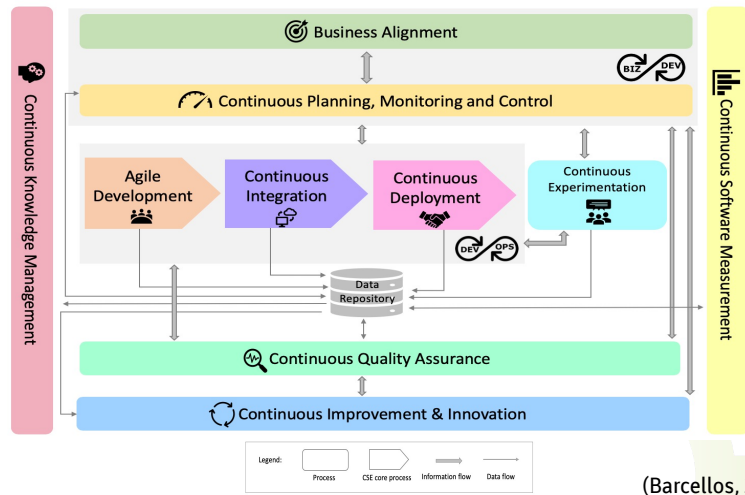


To which processes do the practices relate?



Towards a Framework for CSE

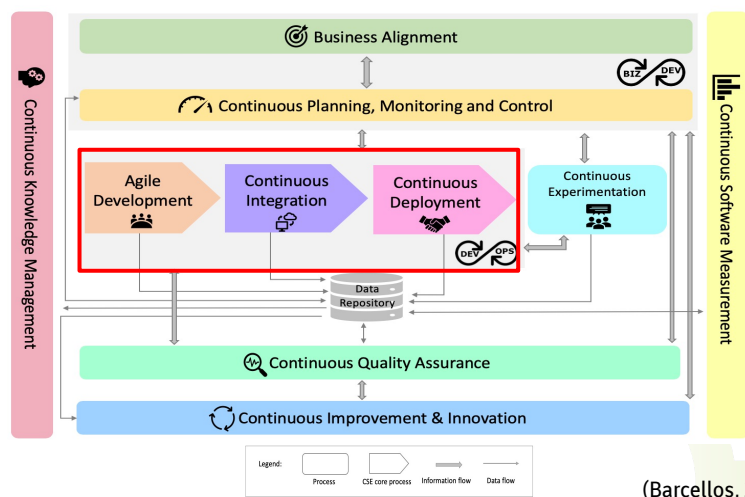
- It identifies CSE processes and the main relations among them by means of information and data flows.



13

Framework for CSE

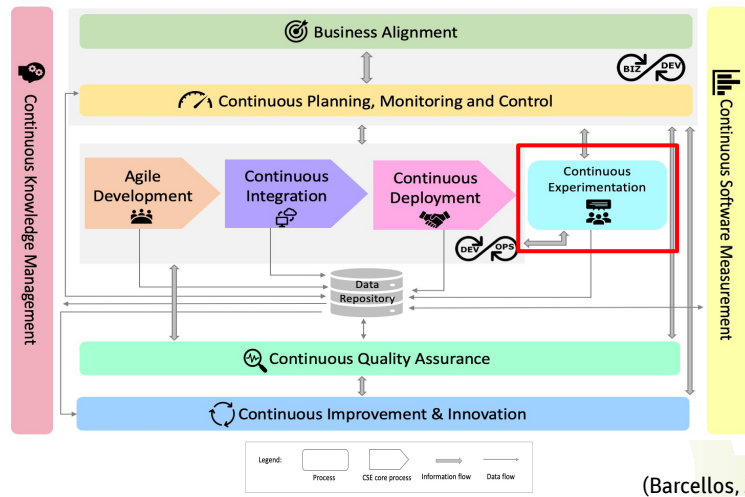
- It identifies CSE processes and the main relations among them by means of information and data flows.



14

Framework for CSE

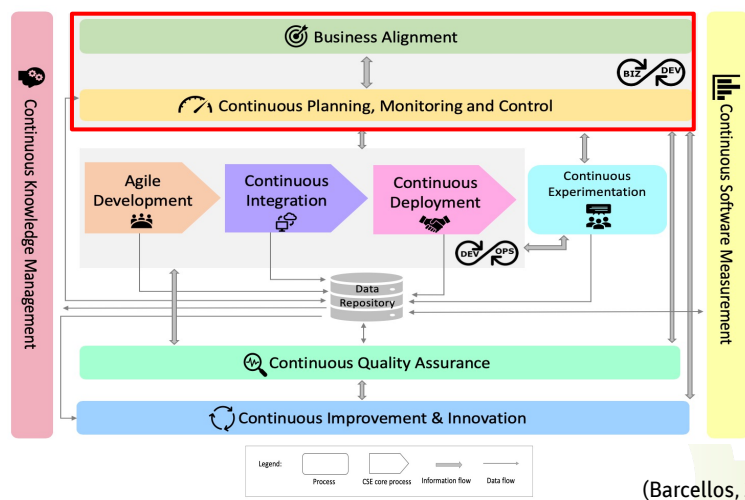
- It identifies CSE processes and the main relations among them by means of information and data flows.



15

Framework for CSE

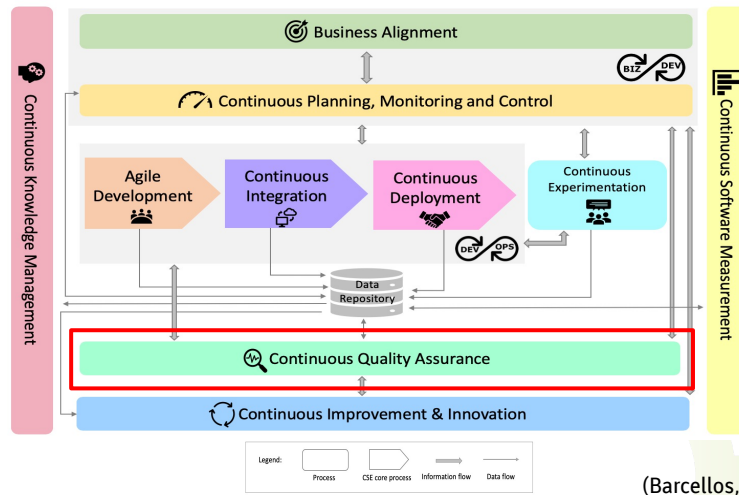
- It identifies CSE processes and the main relations among them by means of information and data flows.



16

Framework for CSE

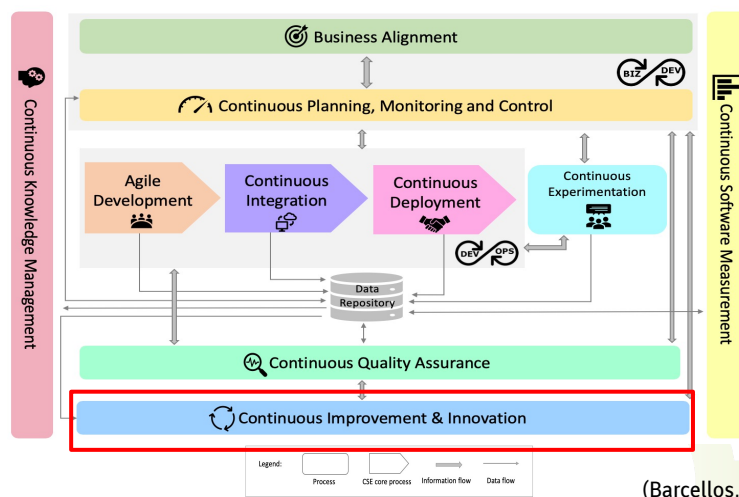
- It identifies CSE processes and the main relations among them by means of information and data flows.



17

Framework for CSE

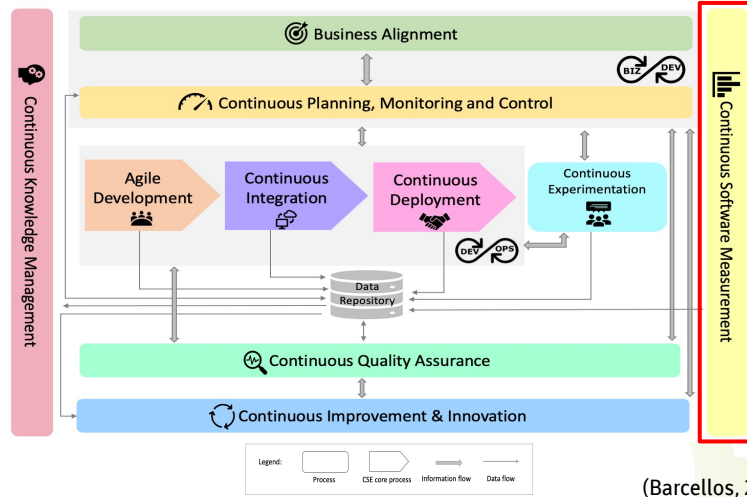
- It identifies CSE processes and the main relations among them by means of information and data flows.



18

Framework for CSE

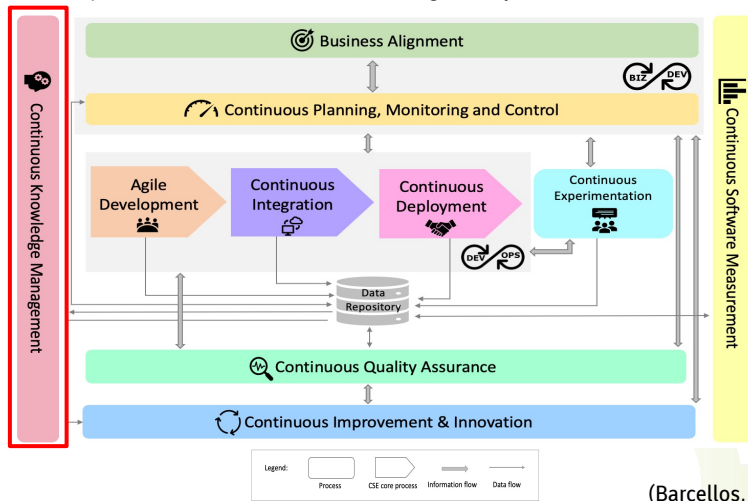
- It identifies CSE processes and the main relations among them by means of information and data flows.



19

Framework for CSE

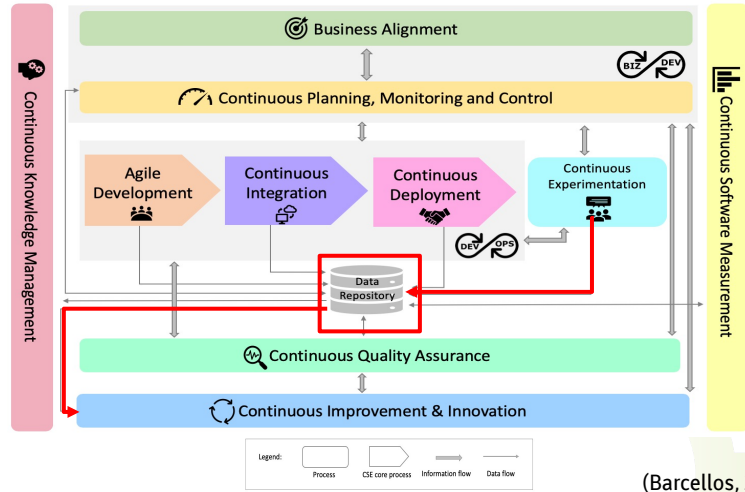
- It identifies CSE processes and the main relations among them by means of information and data flows.



20

Framework for CSE

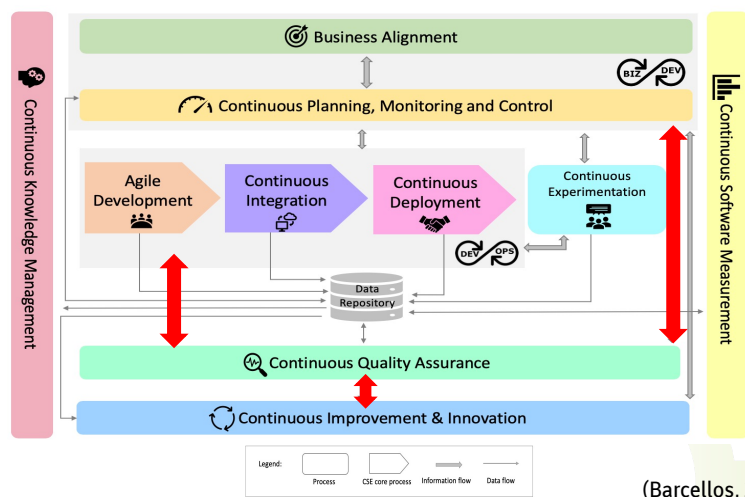
- It identifies CSE processes and the main relations among them by means of information and data flows.



21

Framework for CSE

- It identifies CSE processes and the main relations among them by means of information and data flows.



22

Research Questions and Proposals of Solution

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

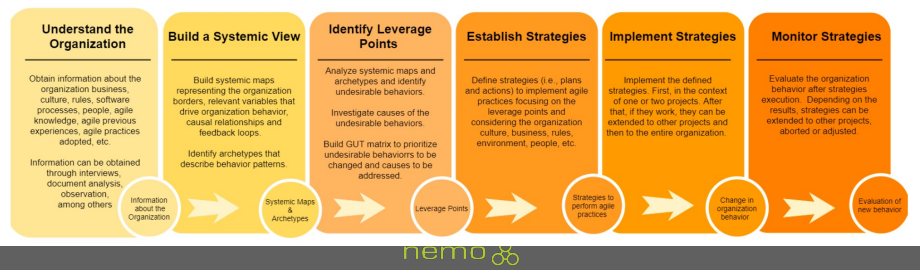
(Barcellos, 2020)

It is necessary to grow knowledge about how CSE processes can be implemented.

Different organizational contexts need to be taken into account because CSE practices must be tailored to fit the organization context.

System-Thinking based process:

(dos Santos Jr. et al., 2020)



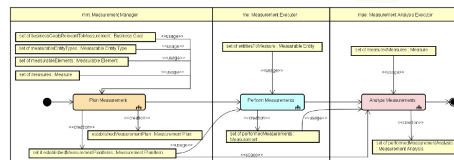
23

Research Questions and Proposals of Solution

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

(Barcellos, 2020)

Use of **task ontologies** to define the processes.



A task ontology clearly represents the process and provides knowledge to answer: (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?

24

Research Questions and Proposals of Solution

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

(Barcellos, 2020)

Establishment of a set of criteria to select tools for the CSE tool chain.



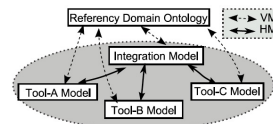
25

Research Questions and Proposals of Solution

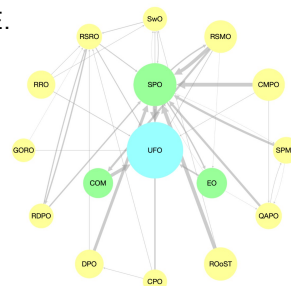
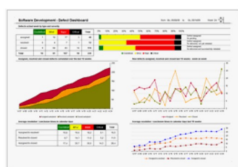
(Barcellos, 2020)

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

RQ5. Which measures can be used in CSE?



Ontology-based solutions for tools integration to provide data to support data-driven software development and decision making in CSE.



nemo

26

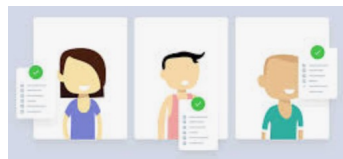
Research Questions and Proposals of Solution

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

(Barcellos, 2020)

Identify methods and techniques to support obtaining user feedback and stimulating user to make explicit his/her implicit impressions.

Extend our data integration and measurement solution (RQ4 and RQ5) to cover user feedback data.



nemo

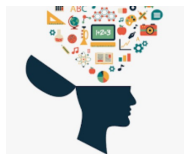
27

Research Questions and Proposals of Solution

RQ7. How to manage knowledge and promote knowledge sharing without creating bottlenecks in the software development process ?

Identify useful knowledge and propose techniques/methods to capture, store, retrieve and use it.

Represent knowledge in such a way that encourages teams to search and retrieve it.



nemo

28

The Road Ahead

The framework provides an overview of a CSE environment:

- It can help practitioners to better make decisions about how 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.

Other questions can be investigated.

New processes can be added.

References

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

Helena H. Olsson, Hiva Alahyari, and Jan Bosch. 2012. Climbing the “Stairway to Heaven” - A Multiple-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.

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.

M. P. Barcellos, “Towards a Framework for Continuous Software Engineering,” in 34th Brazilian Symposium on Software Engineering (SBES 2020), 2020, p. 626–631.

P. S. dos Santos Jr, M. P. Barcellos, and R. F. Calhau, “Am I Going to Heaven? First Step Climbing the Stairway to Heaven Model – Results from a Case Study in Industry,” in 34th Brazilian Symposium on Software Engineering (SBES 2020), 2020, p. 309–318.

Continuous Software Engineering

Monalessa Perini Barcellos

monalessa@inf.ufes.br

Ontology and Conceptual Modeling Research Group (NEMO)
Computer Science Department
Federal University of Espírito Santo (UFES)

