The Journal of Systems & Software 170 (2020) 110732

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Scrum versus Rational Unified Process in facing the main challenges of product configuration systems development

Sara Shafiee [a], Yves Wautelet [b,*], Lars Hvam [c], Enrico Sandrin [d], Cipriano Forza [d]

[a] Department of Mechanical Engineering, Technical University of Denmark, 2800 Kgs. Lyngby, Denmark
[b] Research Centre for Information Systems Engineering (LIRIS), KU Leuven, Warmoesberg 26, 1000 Brussels, Belgium
[c] Department of Management Engineering, Technical University of Denmark, 2800 Kgs. Lyngby, Denmark
[d] Department of Management and Engineering, University of Padova, Stradella San Nicola 3, Vicenza, Italy

ARTICLE INFO

Article history:
Received 15 August 2019
Received in revised form 12 May 2020
Accepted 9 July 2020
Available online 15 July 2020

Keywords:
Product configuration systems (PCSs)
Agile
Scrum
Rational unified process (RUP)

ABSTRACT

Product configuration systems (PCSs) are software applications that enable companies to customise configurable products by facilitating the automation of sales and engineering. Widely used in various industries, PCSs can bring substantial benefits and constitute a fundamental tool for mass customisation. However, serious challenges in PCS development have been reported. Software engineering approaches, such as the rational unified process (RUP) and Scrum, have been adopted to realise high-quality PCSs, but research insights on their use in PCS development are very limited, and their different capabilities to address PCS challenges are almost totally unexplored. This article illustrates the application of RUP and Scrum in PCS development and compares their contributions to addressing PCS development challenges. To perform this comparison, four PCS projects in a company that moved from RUP to Scrum are analysed. The evidence provided suggests that moving from RUP to Scrum has a positive effect in facing organisational, IT-related and resource constraint challenges. The results also highlight worsening knowledge management and documentation, product modelling and visualisation. The findings suggest the adaptation of Scrum for PCS development to reinforce Scrum's knowledge-related capabilities.

1. Introduction

Product configuration systems (PCSs) are software systems that support sales, engineering and production processes (Felfernig et al., 2014; Forza and Salvador, 2006) by incorporating product-related information about features, structures, production processes, costs and prices (Forza and Salvador, 2006). The information required to provide this support is modelled and included in the PCSs during their implementation (Forza and Salvador, 2006; Hvam et al., 2008). Widely used in various industries, PCSs bring substantial benefits to both suppliers of customised products who are seeking an optimal balance between customisation and operational performance (Ardissono et al., 2003; Forza and Salvador, 2002a; Trentin et al., 2012) and to customers who are looking for and co-designing product variants that better match their needs (Kamis et al., 2008; Sandrin, 2017; Sandrin et al., 2017; Trentin et al., 2013). PCSs are therefore recognised as a crucial component in implementing mass customisation (Suzić et al., 2018a,b).

Unfortunately, companies face various challenges in developing and using PCSs (Kristjansdottir et al., 2018; Shafiee, 2017). Among the most important are the knowledge acquisition and product modelling required for complex products and the communication difficulties between domain and configuration experts (Shafiee et al., 2018). A third challenge regards documenting PCSs, which is often not maintained after PCSs become operational because the process is too time-consuming (Haug and Hvam, 2007). These and other challenges increase the risk of PCS project failure (Kristjansdottir et al., 2018) — good management of the PCS development process is required to avoid this (Shafiee et al., 2014).

The rational unified process (RUP), known simply as the 'unified process', is a popular iterative and incremental software development framework (Kruchten, 2007; Shuja and Krebs, 2007). It has inspired the development of PCS projects for years (Hvam et al., 2008; Shafiee et al., 2014), even though its application in PCS projects has not received specific investigation. It is often described as plan-driven, as the entire (iterative) software development life cycle is planned at an early phase of the project based on the units of functionality of the software system under development, known as 'use cases' (Rubin, 2012; Usman et al., 2014; Wautelet et al., 2017). This approach emphasises predictability

* Corresponding author.
E-mail addresses: sashaf@dtu.dk (S. Shafiee), yves.wautelet@kuleuven.be (Y. Wautelet), lahv@dtu.dk (L. Hvam), enrico.sandrin@unipd.it (E. Sandrin), cipriano.forza@unipd.it (C. Forza).

https://doi.org/10.1016/j.jss.2020.110732
0164-1212/

and stability in a project (Boehm, 2012). Plan-driven methods are often used to handle software development projects that support heavy industrial processes in relatively stable business environments and highly critical products (Kruchten, 2007). However, RUP can be heavy — therefore, more agile software development methods have been proposed. Scrum is the most common (Ambler, 2002), offering better communication and faster documentation (Ambler, 2002; Selic, 2009). Previous research has identified shorter development cycles, higher customer satisfaction, lower bug rates and quicker adaptation to rapidly changing business requirements as the benefits of Scrum and other agile methods (Boehm, 2012; Miller and Larson, 2005; Moe et al., 2010). Thus, Scrum has often replaced RUP in several software development projects (Dybå and Dingsøyr, 2008).

Although the literature has signalled the importance of RUP and agile approaches in PCS development projects, to the best of the authors' knowledge, no study has compared these approaches in the case of PCS to inform and assist practitioners in their PCS development efforts. Various studies have focused on the two methods' pros and cons in general software development projects (Cho, 2009; Larman, 2004; Noordeloos et al., 2012; Usman et al., 2014). Other studies have investigated the tailoring/adaptation of agile methods (Campanelli et al., 2018; Campanelli and Parreiras, 2015), and others agile methods' challenges (Drury-Grogan et al., 2017) in general software development projects. However, the authors cannot guess how applicable the results of these studies are to PCS projects due to the numerous differences between PCS projects and other software development projects. For example, determining the scope of the project and plan activities in the early phases of PCS development is a crucial and difficult task due to the complexity of the knowledge characterising these projects (Shafiee et al., 2014). The knowledge modelled in PCSs is extensive and must continually be validated by domain experts (Shafiee et al., 2018) — intense communication between the configuration team and domain experts is therefore vital (Forza and Salvador, 2002b). Not only the knowledge but also the users of this knowledge have different functions (Forza and Salvador, 2006) — consequently, there is a vast range of stakeholders, and the system requirements involve intensive user interaction (Shafiee, 2017; Studer et al., 1998). Specific, detailed and comprehensive documentation and maintenance of product knowledge results are therefore unique in PCS projects (Haug and Hvam, 2007; Shafiee et al., 2017). Due to the differences and diversities in required knowledge, PCSs evolve during project life cycles and are evaluated differently from general software development projects.

Considering the practical relevance of the method chosen to develop PCS, the challenges associated with PCS projects and the paucity of research on the use of RUP and Scrum in PCS projects, this paper provides empirical evidence for using RUP and Scrum during PCS project development. It also compares their applications and effects on PCS challenges. To pursue this aim, the paper considers PCS projects executed in a company that moved from RUP to Scrum in PCS development in 2016. This company, a large engineer-to-order manufacturer that operates globally and is specialised in catalyst production and process plant technology, has considerable experience in PCS projects and is accustomed to professionally managing these projects. Scientifically comparing the impact and results of (software) development methods is inherently very difficult. In addition to the organisational environment's influence on the PCS development process, various internal factors are likely to affect the project's realisation and success. Consequently, the four projects included in the present empirical investigation were selected according to literal and theoretical replication logic. Specifically, these projects are similar because the project team, the type and size of the project, the complexity of the project and the software platform, architecture and integrations are the same. However, the method chosen to manage them (either RUP or Scrum) is different and constitutes the element of variability across projects. The comparison of the four projects is based on archival data as well as on interviews and follow-up meetings with four project members who participated in all four projects. This information is complemented by non-systematic direct observation during project planning and execution performed by two of the authors.

This paper describes and compares RUP and Scrum via the performed activities and challenges of PCS projects. It provides evidence that the advantages Scrum provides in PCS projects are aligned with those reported for general software development projects. However, there are also some weaknesses with Scrum that are particular to PCSs and are related to specific PCS challenges. This new evidence on Scrum and RUP application in PCS projects is valuable for both researchers and practitioners. The results also indicate a direction to develop new ways to face the challenges of PCS implementation and maintenance by adapting the documentation part of Scrum to PCS development.

The remainder of the paper is structured as follows. Section 2 discusses the relevant literature, and Section 3 explains the study's research method. Section 4 describes the characteristics of the case study, and Section 5 discusses the results obtained. Section 6 presents the study's implications for research and practice.

2. Literature background

2.1. Rational unified process (RUP) vs scrum in general SW development projects

RUP is a software development methodology that does not follow the traditional waterfall approach (Usman et al., 2014) but prescribes various incremental iterations to obtain user feedback, which is useful for aligning the software solution with user requirements (Ambler, 2002). Each iteration goes through four phases: inception, elaboration, construction and transition (Ambler, 2002; Kruchten, 2007; Shafiee et al., 2014). In each iteration, sets of activities (called 'disciplines') are performed (Kruchten, 2007). The core disciplines are business modelling, requirements, analysis and design, implementation, testing and deployment; the support disciplines are configuration and change management, project management and environment (Cho, 2009). The majority of each set of activities is performed in one phase, though part is performed in other phases as well (Kruchten, 2007).

Scrum is an agile software development methodology. The agile principles include customer satisfaction, embracing change during development and an active collaboration between experts and software developers (Paetsch et al., 2003). Despite RUP, a Scrum-managed project is typically comprised of short iterations (called 'sprints') lasting from 2–4 weeks to enable rapid feedback from the to-be system's users and stakeholders on the basis of the built releases (Vlietland et al., 2016; Wautelet et al., 2017). The development team fills a so-called product backlog (i.e. a prioritised set of software functions or tasks that must be built or supported) (Rubin, 2012). The Scrum team is responsible for developing a system supporting the elements documented in the sprint backlog (Rising and Janoff, 2000). With the product backlog as the main guidance, the development team first builds solutions for the highest-priority features. Determining priority is done using various factors, even though the potential business value is usually the most important element. When the team is out of resources (e.g. time), the features that have yet to be implemented become of secondary priority. The Scrum master arranges daily meetings, maintains a record of the meetings to gauge the team's

velocity (Cho, 2009), tries to resolve any production problems and coordinates activities between the Scrum team and the rest of the organisation. The product owner (Larman and Vodde, 2013) exclusively represents the customers, eliciting their requirements and sending them a prioritised list called a 'product backlog' (Cervone, 2011).

Both RUP and Scrum are methods for software development that are iterative, prescribe self-managed teams, focus on continuous quality control and learn lessons at the end of an iteration (Collaris and Dekker, 2010). Even though they share several management practices, their scope is different. RUP covers the entire development life cycle, whereas Scrum leaves out the choice of design and implementation techniques and artefacts. More specifically, RUP is a complete method that defines artefacts, roles, activities and supporting tools (Iacob, 2008); Scrum only defines roles, management principles and basic requirements artefacts (Ambler, 2005).

The fundamental difference between Scrum and RUP is that RUP projects are managed in an end-to-end fashion — each iteration and phase are predetermined (in number and length) during the earliest stages of the project (Usman et al., 2014). Iterations theoretically cover each discipline, and a full cycle should be performed before moving on to the next one. In Scrum, the scope of the next iteration (sprint) is determined at the end of the previous one, and the number and length of the following sprints are undetermined until the sprint is effectively tackled. The elements that are the subject of an iteration are also selected in different ways in RUP and Scrum. RUP has a formal risk management approach, dictating that requirements with the highest risk exposure are tackled first; Scrum prescribes a focus on the requirements delivering the highest business value (Appleton et al., 2003; Rubin, 2012; Usman et al., 2014). In addition, requirements representation is different in RUP and Scrum, which has a direct effect on how changes are managed. RUP builds requirements models based on (coarse-grained) use cases (Rubin, 2012; Usman et al., 2014), whereas Scrum uses (fine-grained) user stories (Wautelet et al., 2014). In RUP, changes in requirements are traced back to the scope of a specific use case and tackled within an iteration focusing on that use case. In the Scrum process, a change can be immediately tackled in the form of fine-grained user stories to be addressed in the next sprint (Cho, 2009; Noordeloos et al., 2012). RUP requirements representation is inherently more efficient for the heavy, process-driven applications typically found in industrial environments (Kruchten, 2007), whereas Scrum performs much better for user-driven applications, such as in the service sector (Noordeloos et al., 2012; Rubin, 2012).

Several advantages derive from using the RUP method when compared to pure waterfall approaches: improved governance, regular feedback to stakeholders, improved risk management, implementation of the actual requirements, the ability to early discover whether the architecture of the system to be developed works and continues to meet the changing needs of stakeholders thus mitigating technical risks, and finally developers focus on what really matter (i.e. software development) and not on bureaucracy (Ambler, 2005). RUP also provides high predictability, stability and good quality for large-scale projects (Boehm and Turner, 2005) as well as a systematic mechanism and formalised means of capturing user requirements through elicitation techniques that differ from those associated with agile methods (Boehm and Turner, 2005). The main challenge associated with RUP is that it often exceeds budgetary constraints and is often behind schedule, especially when requirements change (Boehm, 2012). When using RUP, people work in a more isolated manner and with limited communication (Noordeloos et al., 2012). In order to manage and maintain the system, RUP focuses on producing more documentation.

Scrum solves some of these challenges, as it eases coordination and collaboration among multiple teams (Vlaanderen et al., 2011). Better teamwork and better communication result in higher-quality products (Hanakawa and Okura, 2004). Scrum enhances coordination by introducing specific practices involving multiple teams, such as planning sprints and reviews and product refinements (Vlietland et al., 2016). The final success of a project conducted with Scrum strongly depends on the development team (Cardozo et al., 2010), who makes development decisions based on a consensus; this implies that the team has a high degree of control over what is effectively built (Cho, 2009).

Even though Scrum and agility in general have proven beneficial, several organisations are still reluctant to make the switch. Indeed, Cho (2009) highlighted that agile methods (1) have a huge impact on the documentation produced (the latter is considerably reduced) with a team that strongly relies on tacit knowledge, (2) are not yet suitable for safety-critical systems where formal proofs need to be furnished, (3) are not a good fit for software projects where stability is critical, (4) can be used successfully only by trained people with a high degree of freedom and (5) are poorly scalable. Agile development and Scrum have limitations such as their light documentation and poor fit for large-scale projects (Collaris and Dekker, 2010; Usman et al., 2014).

2.2. Differences between general software development and PCS projects

PCS projects present several important differences from other kinds of software development projects (Kristjansdottir et al., 2018) that mainly relate to the special role that product-related knowledge plays.

Knowledge diversity and complexity as well as continual extension and changes inherent to PCS projects during planning make their scoping process challenging. Knowledge diversity includes very different aspects of product knowledge, such as various representations of the knowledge to different stakeholders (e.g. customers and engineers), product features, product components, product production processes, feasible combinations, costs and prices, etc. General software development projects' scopes are often determined differently from PCS because they do not always involve extensive product knowledge (Shafiee et al., 2014).

The second difference is the varying levels of detail in each step of a PCS project. The knowledge modelled in PCSs is considerably diverse and requires domain experts' confirmation (Basili and Weiss, 1984). Thus, modelling tools are introduced to simplify communication between the configuration engineers and domain experts (Forza and Salvador, 2002a). A lack of testing and validation of this knowledge can cause output errors, even in the case of very minor misunderstandings.

The third difference comprises of the highly detailed documentation of PCSs' product-related knowledge (Haug and Hvam, 2007). It should be translated to non-technical language so as to be explicit across the organisation. In PCSs, a great deal of knowledge must be shared among the actors located along the supply chain who are involved in conceiving, building and marketing the customised products.

Moreover, the continuous updates and changes in product knowledge call for an intense knowledge maintenance and documentation process (Felfernig et al., 2000; Friedrich et al., 2014). In contrast, the documentation in general software development projects consists of a summarised explanation of the codes (Coram and Bohner, 2005) and not necessarily the details of the product-related knowledge; thus, the required knowledge for these projects is not constantly updated (Coram and Bohner, 2005).

Table 1
Challenges reported for PCS development projects (Kristjansdottir et al., 2018).

Categories of PCS management challenges	Relative importance	Description	Specific challenges
Resource constraints	Low	Lack of personnel to model the PCS and to gather and provide information, dependency on resources	• Lack of resources • Vulnerability if key personnel leave
Product-related challenges	Low	Challenges in the product range, commonly described as complexity of the product structure and continuous change in products	• Complexity of product structures • Continuous change in product offerings
IT (technical) challenges	Medium	All technical challenges related to IT systems (e.g. software personalisation, user interface design, scope expansion, interaction with software suppliers, functionalities)	• Software development • Systems design for user-friendliness
Knowledge acquisition challenges	High	Difficulties in knowledge gathering and availability of information in the development and maintenance phases	• Difficulty acquiring the correct knowledge • Lack of the requisite knowledge to meet users' and customers' needs • Failure to communicate knowledge in the maintenance phase
Product modelling (knowledge representation) challenges	Medium	Challenges related to formalising the product knowledge and model to be embedded in the PCS	• Complexity due to lack of product range overview • Correctness of specifications generated by the configurator according to the product model • Lack of knowledge related to product modelling
Organisational challenges	Very high	Lack of support from management, resistance to change, allocation of resources	• Lack of support from top management • Resistance to using the PCS • Disagreements about the scope of the configurator

2.3. Challenges of PCS development projects

PCS development projects come with genuine challenges, as recognised by the first academic article that considered PCS (Barker et al., 1989) and then by many subsequent publications (e.g. Aldanondo et al., 2000; Ardissono et al., 2003; Ariano and Dagnino, 1996; Forza and Salvador, 2002a; Heiskala et al., 2007). Kristjansdottir et al. (2018) provide a detailed discussion of the main PCS development challenges; Table 1 contains a synthesis of their main results.

2.4. RUP and scrum in PCS development projects

Research on the use of RUP versus Scrum to manage PCS development projects is lacking. This is evidenced by the absence of journal articles in Scopus and Web of Science databases when the search [(RUP OR 'Rational Unified Process') AND Scrum AND configur*] is performed in these databases in the title, abstract and keywords. This gap is the result of a general paucity in research on both RUP and Scrum in managing PCS projects.

Focusing on just RUP in PCS projects, only one relevant journal article was found after examining the results provided by the search string [(RUP OR 'Rational Unified Process') AND configur*] performed in the same fields and databases as mentioned above. By considering the references cited in this article and the publications that cite this article, only four publications emerged that dealt, to a limited extent, with RUP in PCS projects. Hvam et al. (2008) proposed a procedure to develop PCSs by building on different methods, including RUP. Hvam et al.'s (2008) PCS development procedure consisted of seven stages: (1) development of the specification processes, (2) analysis of the product range, (3) object-oriented modelling, (4) object-oriented design, (5) programming, (6) implementation and (7) maintenance and further development. Focusing only on the inception of a PCS development process, Shafiee et al. (2014) presented an ad hoc framework inspired by RUP to support the initial PCS scoping process; they proved its usefulness in supporting early clarification and scoping. Shafiee et al. (2016) used five case companies to investigate applying Hvam et al.'s (2008) procedure. Shafiee et al. (2018) found several RUP tools to be beneficial while scoping PCS projects. However, none of these publications presented a full application or the pros and cons of using RUP in PCS development projects.

Regarding the use of Scrum to manage PCS development projects, no journal articles emerged after examining the results yielded by the search string [Scrum AND configur*], which was performed in the same fields and databases as mentioned above. This is quite surprising, given the increasing attention that researchers are paying to Scrum on the one side and PCS on the other. This could be understandable if Scrum was not used for PCS development projects, but some companies made or are making a transition from RUP to Scrum to manage their PCS development projects, and many others are debating whether it is effective to use Scrum for PCS projects.

Since PCS development projects have important differences from general IT projects (see Section 2.3), it is impossible to infer that the research findings obtained about using RUP and Scrum in general IT projects also hold for PCS projects. This would be not a large problem if PCS projects were easy to manage and highly successful, but there are many significant challenges that undermine their success (see Section 2.4). Consequently, the limited research on the use of RUP and Scrum in PCS development create several open questions that are important for practice: Do RUP and Scrum work for PCS projects? Which one of these works better for PCS projects? Do RUP and Scrum need adaptations in order to work for PCS projects? Do RUP and Scrum address PCS challenges? Which PCS challenges are alleviated/worsened by RUP? Which PCS challenges are alleviated/worsened by Scrum?

There is a need to further investigate the possibility of using RUP and Scrum in PCS development.

3. Research aim and method

3.1. Research aim

This article intends to at least partially cover this paucity of research by answering the research question, *What are the differences between RUP and Scrum with respect to how PCSs are developed and their impact on PCS development challenges?* Doing this not only provides indications on the application of RUP and Scrum in PCS projects, but it also offers evidence about whether Scrum can perform better than RUP in some of the challenges with 'traditional' PCS development identified in the literature.

3.2. The case study method

A case study was designed in accordance with the initial stage of the research on the present topic (Edmondson and Mcmanus, 2007). Case studies permit researchers to identify and describe key variables, uncover linkages between variables and understand the 'whys' behind these relationships (Voss et al., 2002; Yin, 2009). This type of empirical enquiry investigates a contemporary phenomenon within its real-life context (Yin, 2009) and is therefore a particularly good fit for software engineering research (Runeson et al., 2012; Runeson and Höst, 2009). Case study research enables deep observation of the phenomenon under investigation, and for a given set of available resources, fewer cases allow for deeper observation (Voss et al., 2002).

Multiple case study was chosen because it allows for the observation of different life cycles – that of RUP and that of Scrum – in PCS projects and their impacts on PCS challenges. Specifically, the research is based on a multiple case study of four PCS development projects (i.e. cases) within the same company.

3.3. The company

The extreme case decision rule was followed to select the company (e.g. Benbasat et al., 1987; Yin, 2009) because the selected company represents a highly relevant and rare context for this study's enquiry. Specifically, the selected organisation has considerable experience in PCS projects and is accustomed to professionally managing these projects. In 2016, it moved from RUP to Scrum to develop its PCS. That this company has experience with both software development approaches makes it quite a unique case that suits the objectives of the present study particularly well.

The company is a large engineer-to-order manufacturer that operates globally and specialises in catalyst production and process plant technology. It is an industrial partner involved in one of the university research studies in the field of PCS projects. This relationship enabled two members of the research team to observe, at a different level but not in a systematic way, RUP-guided and Scrum-guided PCS projects, including the considered ones, as they were happening. Both the company and the researchers wished to understand how RUP and Scrum work in PCS development. Consequently, the company's top management supported the present study by granting complete access to its data sources and guaranteeing the participants' commitment to the project.

Notably, contextual variables may affect the challenges associated with PCS projects and the successful application of RUP and Scrum methods. Although performing the study in one organisational setting has the disadvantage of limiting the generalisability of the results, it has the advantage of controlling for a number of contextual variables – such as organisational characteristics and culture – that could confound the results and lead to spurious relationships.

3.4. The unit of analysis and the choice of PCS projects

The present study's unit of analysis is the PCS project. The enquiry mainly concerns the challenges that a PCS project encounters when it is managed with RUP or Scrum. Even within the same company, the challenges that a PCS project encounters, as well as the successful application of RUP and Scrum methods, may differ because different software is used, because different degrees of application complexity are involved, because projects have different sizes, because of differing levels of experience with the software development resources, because PCS projects have different degrees of predictability, different requirements, etc. All these factors may influence the effects of RUP and Scrum on PCS projects. However, the retrospective selection of cases allowed the researchers to mitigate these negative effects because 'retrospective cases allow for more controlled case selection' (Voss et al., 2002, p. 202). Therefore, PCS projects that were as similar as possible in all these respects were selected by considering projects on the most popular catalysts at the company. In this way, many projects were similar in terms of product technology and key characteristics. The researchers chose projects in which the only relevant feature that changed was the method used — either RUP or Scrum.

Four of the company's projects were selected as cases following literal and theoretical replication logic (Runeson and Höst, 2009; Voss et al., 2002; Yin, 2009). The first two PCS projects (Projects 1 and 2) were developed using RUP; the other two (Projects 3 and 4) were developed using Scrum. The PCS development team members had 2–10 years of experience working with both methods. The four projects shared the following characteristics: (1) origin in the same company, (2) a PCS for products with almost the same level of complexity (medium to medium–high), (3) use of either RUP or Scrum for PCSs, (4) potential access to management and senior experts at the companies, (5) development of all the selected PCSs in one specific software platform, (6) similar requirements, (7) similar users (engineers), (8) similar development team and the involvement of similar tasks and (9) knowledge of similar setups, software architecture and integrations.

3.5. Time-phased overview of the research

Fig. 1 uses a workflow to depict the research activities in the present study. The PCS development team (i.e. the one working on the various PCS projects) had been established seven years prior to the start of the present study; it had used RUP for five years and has used Scrum for two years (see the timeline underneath the workflow in Fig. 1).

First, two activities were performed in parallel: *(1) Literature review.* As explained above, the research team consulted relevant sources to build up and present background knowledge on the subject. *(2) Analyse archived documents.* The research team analysed and discussed the available documents on the different PCS development projects that the case organisation had analysed and archived. These two parallel activities took the research team five months to complete, and together, they led to the creation of a questionnaire that was used to conduct study interviews.

Subsequently, the following research activities were completed sequentially: *(3) Conduct interviews*. Each interviewee was interviewed separately from the others. In each interview, the entire questionnaire was filled in, and all four projects were considered. In total, the interview activities took about four months to complete. *(4) Analyse the results of the interviews.* The research team then tabulated, discussed and analysed all the interview and document results. *(5) Confirm the results in the feedback meeting.* The research team presented the results to the

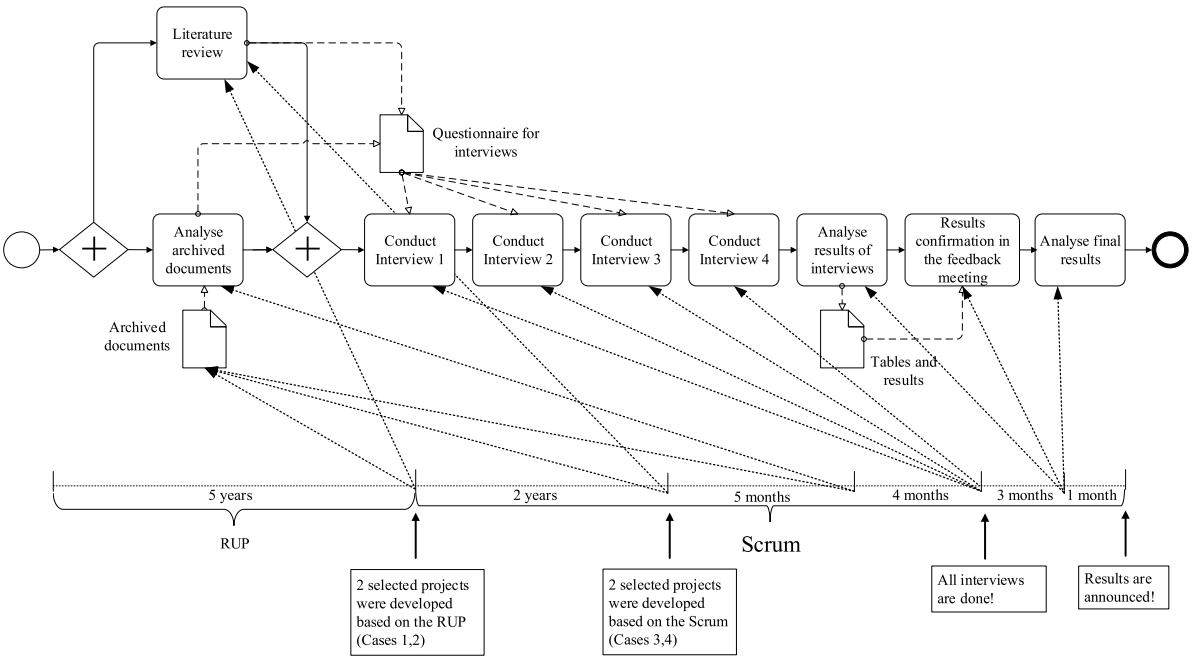


Fig. 1. Time-phased study activities.

interviewees and other PCS development professionals in the case company to share the findings and to get feedback to enhance the results' validity. *(6) Analyse the final results.* This last activity involved analysing the final comments, remarks and issues the professionals at the feedback meeting raised. Together with the two previous activities, this took three months to complete.

3.6. Data collection

Data collection was performed using multiple data sources via non-systematic direct observation, archived document analysis and semi-structured interviews. Two members of the research team performed the direct observation. The first observed the considered PCS projects as well as other PCS projects at a high level of interaction with the company, learning about the context and the reasons that motivated the company to move from RUP to Scrum. The second observed all four considered projects and other PCS projects (eight RUP and five Scrum); the researcher also participated in several meetings and activities. For both observers, their work was significant in terms of the information acquired, but it was not systematic.

3.6.1. Archived documents

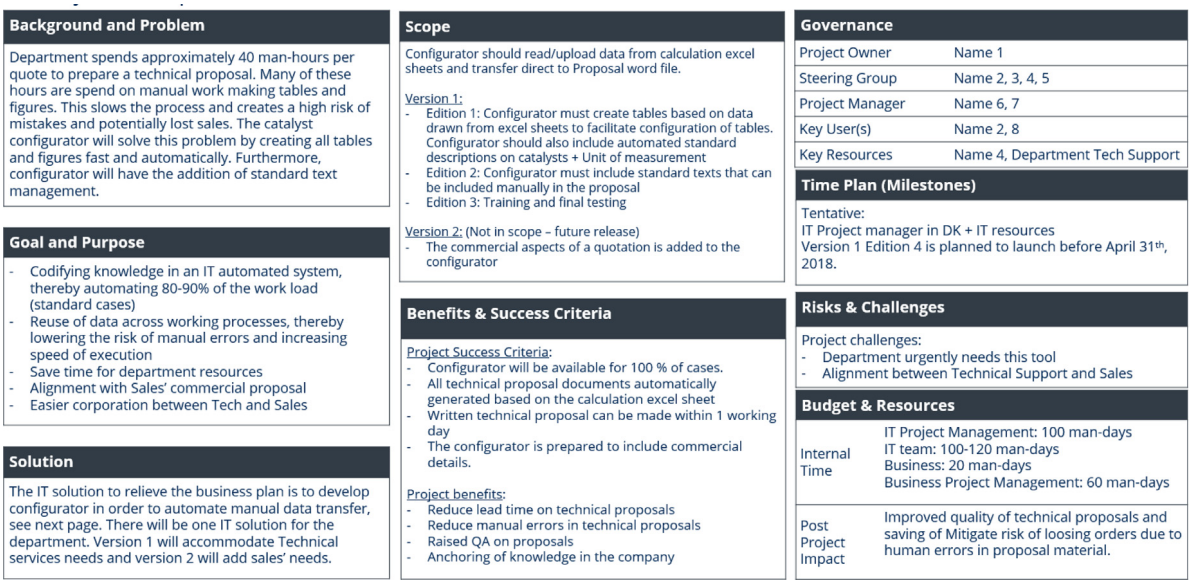
All the available documents generated during the four projects were collected and studied. Specifically, for the two RUP-based PCS projects, the following documents were analysed: project description for the initiation phase (including budget, stakeholder analysis, risks and involved resources), meeting minutes and discussions, all the documents relevant to the analysis in different phases (use case diagrams, AS-IS and TO-BE flowcharts, the product variant master (PVM), unified modelling language (UML) class diagrams and Class-responsibility-collaboration (CRC) cards) and maintenance charts. For the two Scrum-based PCS projects, the following documentation was analysed: sprint recovered from Jira (a software platform that allows users to track all the Scrum management artefacts such as backlogs, task prioritisation, and task allocation), including all the stories, assigned resources, story points, dependencies, acceptance criteria, testing story points, feedback and comments, etc.; and the meeting minutes for each project, including daily Scrum, print planning, sprint review (lessons learned, what went well, what can be improved) and feedback meetings. These documents were used as literal recordings of past events, constituting how the researchers observed the PCS development process as it happened. Understanding of these documents was greatly facilitated by the insights gained through the non-systematic direct observations performed by two of the researchers. Several short phone calls and short meetings were conducted to clarify confusing information.

The in-depth reading of the documents allowed the researchers to understand the context of the projects and build an identity card for them. It also allowed them to ascertain what artefacts were produced and used during the RUP- and Scrum-based projects. Noticeably, the archived documents were not originally developed with the intention to provide data for case study research — the case organisation required them in order to document the projects. On the one hand, this limited the researchers' capability to collect information, and on the other hand, it did not influence the knowledge documentation and project documentation of the four considered projects.

The identity cards we build were quite extensive. In order to check them at the company and use them during interviews to have facts on hand as well as our reconstruction of each PCS case, we split them into different figures, sub tables and slides. In Fig. 2, we report the most important slide, which contains the scope of the project, team members, and time investments for the Project 1 as planned before the launch of the project. For RUP-based PCS projects, we added also PVM diagrams (an excerpt from Project 2 is reported in Fig. 3). For Scrum-based projects, we included also the list of the user stories with all the details generated from the Jira system to consider the exact time and resources (Fig. 4 report this for Project 4).

3.6.2. Interviews

In accordance with the research objectives, the interviews were designed to systematically compare the RUP and Scrum approaches with respect to PCS project challenges. The interview process was semi-structured and involved experienced researchers and experienced PCS project stakeholders.



Background and Problem

Department spends approximately 40 man-hours per quote to prepare a technical proposal. Many of these hours are spend on manual work making tables and figures. This slows the process and creates a high risk of mistakes and potentially lost sales. The catalyst configurator will solve this problem by creating all tables and figures fast and automatically. Furthermore, configurator will have the addition of standard text management.

Goal and Purpose

- Codifying knowledge in an IT automated system, thereby automating 80-90% of the work load (standard cases)
- Reuse of data across working processes, thereby lowering the risk of manual errors and increasing speed of execution
- Save time for department resources
- Alignment with Sales' commercial proposal
- Easier corporation between Tech and Sales

Solution

The IT solution to relieve the business plan is to develop configurator in order to automate manual data transfer, see next page. There will be one IT solution for the department. Version 1 will accommodate Technical services needs and version 2 will add sales' needs.

Scope

Configurator should read/upload data from calculation excel sheets and transfer direct to Proposal word file.

Version 1:
- Edition 1: Configurator must create tables based on data drawn from excel sheets to facilitate configuration of tables. Configurator should also include automated standard descriptions on catalysts + Unit of measurement
- Edition 2: Configurator must include standard texts that can be included manually in the proposal
- Edition 3: Training and final testing

Version 2: (Not in scope – future release)
- The commercial aspects of a quotation is added to the configurator

Benefits & Success Criteria

Project Success Criteria:
- Configurator will be available for 100 % of cases.
- All technical proposal documents automatically generated based on the calculation excel sheet
- Written technical proposal can be made within 1 working day
- The configurator is prepared to include commercial details.

Project benefits:
- Reduce lead time on technical proposals
- Reduce manual errors in technical proposals
- Raised QA on proposals
- Anchoring of knowledge in the company

Governance

Project Owner	Name 1
Steering Group	Name 2, 3, 4, 5
Project Manager	Name 6, 7
Key User(s)	Name 2, 8
Key Resources	Name 4, Department Tech Support

Time Plan (Milestones)

Tentative:
IT Project manager in DK + IT resources
Version 1 Edition 4 is planned to launch before April 31th, 2018.

Risks & Challenges

Project challenges:
- Department urgently needs this tool
- Alignment between Technical Support and Sales

Budget & Resources

Internal Time	IT Project Management: 100 man-days IT team: 100-120 man-days Business: 20 man-days Business Project Management: 60 man-days
Post Project Impact	Improved quality of technical proposals and saving of Mitigate risk of loosing orders due to human errors in proposal material.

Fig. 2. Excerpt from the slides used to show a PCS project identity card to the company.

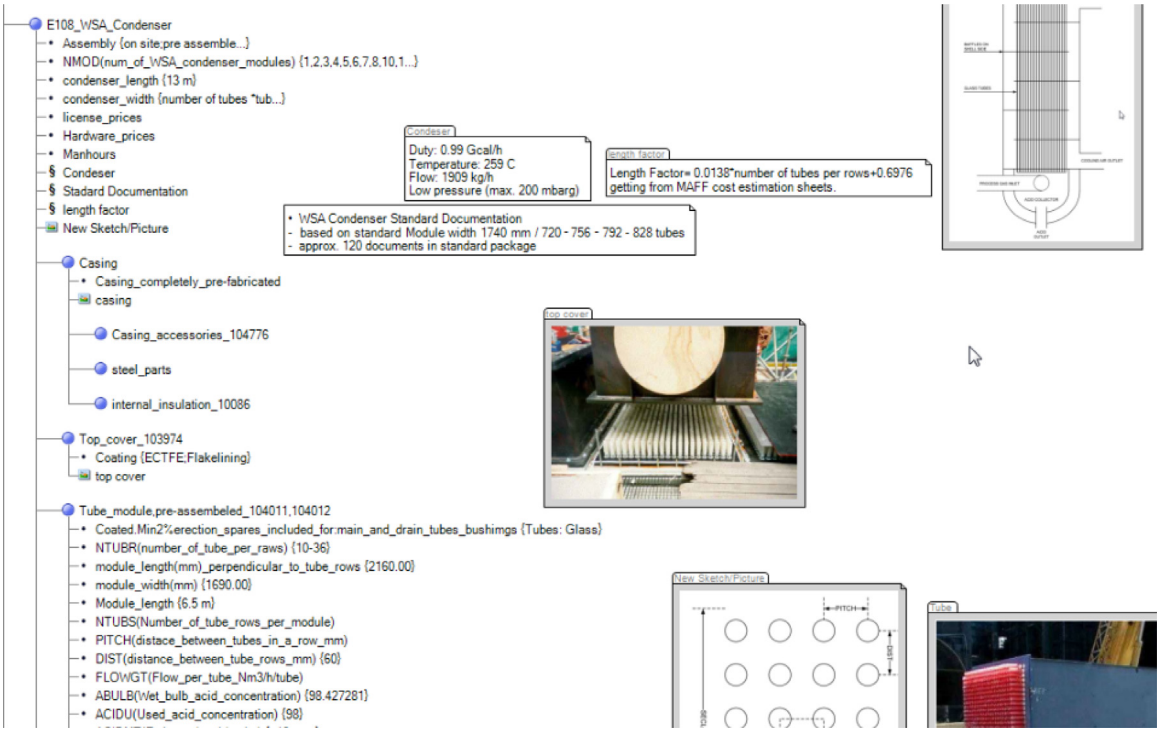


Fig. 3. Excerpt from a PVM included in the identity card of a RUP-based PCS project.

3.6.2.1. The interviewer team. The team consisted of three authors of the paper including one specialist in software development life cycles, one specialist in PCS and one specialist in both domains. Having in-depth knowledge of the RUP and Scrum PCS development processes enabled the team to prompt the interviewees to elaborate upon and justify their answers.

3.6.2.2. The interviewed PCS development teams. The interviews were conducted between 2017 and 2018 after the case organisation completely abandoned RUP in 2016 and immediately adopted Scrum. Interviewee selection was based on several criteria, including similarities and differences in controlling the confounding factors and collecting different point of views caused by different experiences.

The four selected interviewees were the only personnel members who were present in all four PCS projects, which enabled them to answer every question about the four projects. They all had years of experience in PCS development using both RUP and Scrum, and they had all experienced the case company's transition from RUP to Scrum. Thus, they could critically compare their experiences with RUP and Scrum in the PCS projects.

They also had different backgrounds and differing years of experience with PCSs, IT projects and Scrum; hence, they had different perspectives, expectations and concerns regarding Scrum and RUP. They all had years of experience (4, 6, 7, and 12 respectively) in PCS development, they all used both RUP and Scrum and they had all experienced the case company's transition from

T	Key	Summary	Assignee	Reporter	Task importance	Status	Resolution	Created	Updated	Worklog Author	Worklog Time	Time Spent (h)
Story	CPQ-265	step1-Basic info Part1	Name C	Name A	Major (2)	Done	Done	21/Sep/2017	03/Jan/2018			
Story	CPQ-266	Excel mapping for Optimal loading	Name E	Name A	Major (2)	Done		21/Sep/2017	26/Feb/2018			
Story	CPQ-267	Excel mapping for Diesel loading	Name E	Name A	Major (2)	Done	Done	21/Sep/2017	21/Feb/2018	Name C	05/Dec/2017 09:42	0.13
			Name E	Name A						Nanme D	16/Feb/2018 12:53	1.00
			Name E	Name A						Nanme D	21/Feb/2018 11:00	0.50
Story	CPQ-269	coding Diesel sheets-Operation conditions	Name E	Name A	Major (2)	Done	Done	21/Sep/2017	31/Jan/2018	Nanme D	29/Jan/2018 07:22	1.00
			Name E	Name A						Nanme D	24/Jan/2018 07:23	2.00
			Name E	Name A						Nanme D	23/Jan/2018 07:23	2.00
			Name E	Name A						Nanme D	31/Jan/2018 07:55	1.00
Story	CPQ-270	coding Diesel sheets-Feed stock properties	Name E	Name A	Major (2)	Done	Done	21/Sep/2017	29/Jan/2018	Nanme D		
Story	CPQ-271	coding Diesel sheets-Catalyst performance	Name E	Name A	Major (2)	Done	Done	21/Sep/2017	31/Jan/2018	Nanme D	23/Jan/2018 07:26	1.00
				Name A						Nanme D	31/Jan/2018 07:55	0.67
Story	CPQ-272	word template documetation-catalyst performance	Name D	Name A	Major (2)	Done		21/Sep/2017	26/Feb/2018			
Story	CPQ-326	word template documetation-Feed properties	Name D	Name A	Major (2)	Done		09/Nov/2017	22/Feb/2018			
Story	CPQ-327	word template documetation-Operating conditions	Name D	Name A	Major (2)	Done		09/Nov/2017	26/Feb/2018			
Story	CPQ-328	adding all the variables needed-modelling	Name D	Name A	Major (2)	Closed	Duplicate	09/Nov/2017	31/Jan/2018			
Story	CPQ-329	units of measurement options	Name D	Name A	Major (2)	Done	Done	09/Nov/2017	16/Feb/2018	Nanme D	16/Feb/2018 08:04	0.17
Story	CPQ-330	finalizing the word document	Name D	Name A	Major (2)	Done		09/Nov/2017	07/Feb/2018			
Story	CPQ-353	NT: Support and understand the logic with tables for TK catalyst	Name D	Name B	Major (2)	Done	Done	29/Nov/2017	23/Feb/2018	Name E	02/Feb/2018 13:04	2.00
										Name E	05/Feb/2018 08:30	5.00
										Name E	06/Feb/2018 04:41	1.00
										Name E	16/Feb/2018 13:35	3.00
										Name E	16/Feb/2018 13:36	1.00
										Name E	20/Feb/2018 09:21	1.00
										Name E	22/Feb/2018 08:18	5.00
Story	CPQ-383	Loading method table	Name D	Name B	Major (2)	Done	Done	13/Dec/2017	31/Jan/2018			
Story	CPQ-384	Catalyst requirements - part 1	Name D	Name B	Major (2)	Done	Done	13/Dec/2017	31/Jan/2018			
Story	CPQ-385	Catalyst requirements - Part 2	Name D	Name B	Major (2)	Done	Done	13/Dec/2017	31/Jan/2018			
Story	CPQ-403	step1-Basic info Part 2	Name E	Name B	Major (2)	Done	Done	03/Jan/2018	23/Jan/2018	Nanme D	23/Jan/2018 06:24	0.50
Story	CPQ-404	word template mapping - Loading method	Name E	Name B	Major (2)	Done		03/Jan/2018	26/Feb/2018			
Story	CPQ-405	word template mapping - Catalyst requirements	Name D	Name B	Major (2)	Open		03/Jan/2018	26/Feb/2018			
Story	CPQ-454	Update the default values	Name D	Name B	Major (2)	Done		07/Feb/2018	26/Feb/2018			

Fig. 4. Excerpt from the list of user stories included in the identity card of a SCRUM-based PCS project.

RUP to Scrum. More precisely, the first interviewee had 10 years of experience as software developer plus 4 years of experience working with PCS projects and she was a certificate Scrum master with 5 years of experience in Scrum. The second interviewee had 6 years of experience as a configuration engineer at the same company with 3 years of experience working with Scrum. The third interviewee was a software developer with 7 years of experience plus 7 years of experience working with PCS projects and a scrum expert in the last 3 years. The last interviewee was the group leader who was a configuration engineer with 12 years of experience and more than 4 years of experience working with Scrum. They played different roles in the PCS teams, ranging from technical developer and project manager to tester and domain expert. The roles may have varied from project to project, but overall, in the Scrum projects, they were product owners, business analysts, requirements engineers, Scrum masters, software designers, developers and testers. Hence, they could share their varying perspectives, allowing the researchers to consider different PCS challenges from different angles.

3.6.2.3. The questionnaire. In order to explore the differences among RUP and Scrum in facing PCS challenges, a set of statements were developed based on the literature and archival data analysis (see Column 1, Table 4 in the Results section). The level of agreement with each statement was measured on a 5-point Likert scale (see Columns 3–7 of Table 4). Specifically, the interviewees were asked to evaluate the contributions of Scrum versus RUP with respect to specific aspects of the six main PCS challenge categories (see Table 1). In order to facilitate the respondents and not introduce differences that could affect comparisons among the answers to different statements, each statement was constructed as 'Scrum …when compared to RUP'. The entire research team thoroughly discussed each question.

3.6.2.4. The interview process. The semi-structured interview began by presenting the reconstructed identity card for each of the four projects. While going through these, further information on each project was solicited to improve the overall picture of each PCS project. This process of going through each project in detail prompted respondents to recall previous projects in order to prepare them for their comparison questions. As a second step, the interviewees filled in all the closed Likert-scaled questions on the questionnaire. Notably, the questionnaire was emailed to the interviewees before the interviews. If the interviewees felt the need to justify their answers, the interviewer/s recorded the justifications and asked for further detail if necessary. Therefore, the questions were written and asked such that the respondents were free to add comments or opinions (Hollway and Jefferson, 2000). Once the respondents filled in the questionnaires, the interviewers asked them to justify any answers they had not previously commented upon. Finally, the interviewers asked further questions about the impact of RUP and Scrum on PCS challenges that corresponded to the interviewees' reflections on the PCS project comparisons.

Each interview involved a single respondent addressing all four projects and all the planned questions. The synthesis of the answers to the open-ended questions can be found in the last columns of the questionnaire in Table 4. The open-ended portion of the interview not only facilitated the collection of background information to enhance the richness of this comparative study (Yin, 2009), but it allowed the researchers to understand why interviewees considered Scrum to be superior/inferior to RUP. The respondents explained what the situation was like before with RUP and how it was currently with Scrum, describing some specific facts about the tools used in RUP PCS projects. Their scores on the answers they gave regarding case projects, explaining why they provided a given answer and digging into the underlying reasons informed the results. For each interview, a researcher reported the interview answers in a text file and returned the file to the respondent to check and confirm the interview content.

3.7. Data analysis

As anticipated in Section 3.5 and depicted in Fig. 1, data analysis were performed in various phases of the present research. Each analysis built on the results of previous analysis and on the additional data gathered since the last analysis. Both qualitative and quantitative data were structured into tables to identify regularities (as recommended in Runeson and Höst, 2009) and to communicate similarities and differences among cases.

An initial analysis was done on the archival documents collected before the questionnaire design. Two of the researchers independently went through all the documents and identified all the data relevant for understanding each case. They inserted these data in tables, thus building an identity card for each PCS project

that included project context (organisational and team culture), scope, duration, number and roles of employees involved, number of users for the solution, used or produced artefacts, applied theories, planning approaches, meeting disciplines, etc. Subsequently, the identity cards were compared, and their differences were resolved by involving a third researcher. Phone calls were made at this stage to collect missing information and to clarify doubts. To facilitate cross-case comparison, each case was described as phases (sets of activities related to a common theme) that were similar for RUP and Scrum. The subsequent cross-project comparison considered phase-by-phase performed activities and the roles involved; they continued by comparing the PCS projects on aspects that were possibly related to PCS challenges. This analysis produced a first description of the four projects, helped design a focused explorative questionnaire and identified specific aspects to investigate in an open way in the interview phase.

A second set of analysis was performed in parallel with the interviews. The identity cards were used at the beginning of each interview to help the interviewed remember the case projects and to gather additional data on each project, enriching its description. Consequently, the cross-case comparison obtained with the first analysis was further and further refined. However, this refinement process enriched but did not significantly change the previously obtained picture, thus increasing the researchers' confidence in the obtained results. The analysis performed in parallel with the interviews considered the closed questions from the questionnaire, the related explanations provided by the interviewed PCS team members and other answers to open questions (to address comments, extra explanations and reasons) related to identified issues during the analysis performed before the interviews as potentially related to PCS challenges. The answers to the open questions and the discussions about the answers to the closed questions were coded, cross-tabulated and analysed to find patterns and shared explanations. These analysis informed the interviewers' understanding, enabling them to provide additional new stimuli during discussions in subsequent interviews. After concluding the interviews, the various results on the influence of RUP and Scrum on PCS challenges were consolidated.

Finally, feedbacks from presenting the results to the interviewees and other PCS professionals at the case company were considered. The comments and issues they raised led the researchers to deepen some explanations, and in a couple of cases, partially modify them. At this point, the researchers were confident of the results.

4. Results

4.1. How PCS projects were performed with RUP and Scrum

The four projects had some small differences in their considered products: they were all the most popular catalysts at the case company, but they differed regarding product size. They also presented some differences in their complexity. Consequently, the number of product attributes considered in the corresponding PCS projects varied from 1300–2500. The number of constraints inserted in the corresponding PCS varied from 800–2000.

Project duration also differed. The total duration of a project is the time from when the decision has been made to launch it to when it is completed. The net duration of the project is the time from the actual start of project activities to when the project has been closed, removing all the waiting times and decision pauses. However, the main differences between their duration is neither to be imputed to the differences in products and complexity involved nor on some different difficulties encountered during their execution. The comparison of the four projects showed small differences between Projects 1 and 2 on the one hand and between Projects 3 and 4 on the other. In contrast, there were big differences in duration time between projects 1 and 2 versus Projects 3 and 4; these differences were due to the methods used (i.e. RUP or Scrum).

One key difference between RUP-based and Scrum-based PCS projects concerned the number of iterations: 2–3 in the first RUP projects and 9–10 in the second Scrum ones. RUP-based PCS projects were organised in big revisions to develop and test their deliverables, with larger projects organised via more revisions. Scrum-based projects were organised in fixed sprints, with larger projects executed in more sprints.

What follows is a presentation of similarities between Projects 1 and 2 (RUP-based) and similarities between Projects 3 and 4 (Scrum-based). This description depicts what RUP-based and Scrum-based PCS projects look like, which is not currently available in the literature.

4.1.1. RUP-based projects

4.1.1.1. Project values. In the two RUP projects, the main guiding values were systematic progression and risk reduction.

4.1.1.2. Roles and main tasks. The project roles included domain experts (two in Project 1, three in Project 2), end users (varied and increased from the scoping to the implementation phase of the project: 5–8 in Project 1, 8–10 in Project 2), stakeholders (10–13 in Project 1, 15–17 in Project 2), a product owner, a project manager and a development team composed of two IT professionals (a configuration engineer and a developer). In these two projects, the domain experts were chemical engineers from the sales department. These experts knew the types, the characteristics, the functioning and the production process of the catalysts. They were the source of product knowledge inserted into the PCS, both from the customer and the technical point of view. These experts were responsible for communicating the product knowledge and their system requirements and verifying and validating the knowledge inserted into the PCS. The end users were the stakeholders who would be the potential users of the system — in these projects, few were the domain expert resources who provided knowledge. The rest of the end users, who were not directly considered to be project resources, were not supposed to but were allowed to provide the team with functional requirements and feedback. The stakeholders included all the domain experts, end users and top managers who were directly or indirectly involved in these projects. The stakeholders attended steering committee meetings to discuss their requirements and feedback. The product owner was responsible for modelling the product knowledge inside the configurator. The project manager was responsible for all the planning, scoping, product and process analysis and knowledge management. The configuration engineer modelled and tested the configurator. Finally, the developer integrated PCS with other IT platforms (i.e. ERP, CAD, CRM and simulation systems) and tested the integration.

4.1.1.3. Task assignment. The assignments for the IT professional were defined in the beginning of each project, with no flexibility to assign tasks to another resource if needed. This criterion sought to minimise uncertainty and increase efficiency in the task execution, as each person was dedicated to activities aligned with their specialisation. Domain experts were mainly used in a sequential way — they were usually busy with other tasks related to their regular duties. The parallel use of IT professionals and domain experts was not planned but occurred as needed.

4.1.1.4. Activity organisation. Project planning was detailed in the very beginning and subsequently neither re-planned nor further

detailed. Initially, the projects were divided into different versions or releases[1] (with bigger projects broken down into more versions than the smaller projects). In a PCS project, a version can be a part of a product, a full product family, a category/group of similar product variants, etc. For example, the PCS of Project 2 was divided into three main parts corresponding to the categories of the catalyst type. The core of a project was developed in the first version — additional knowledge to complete the project was added in the subsequent versions. For each version, all the knowledge acquisition was performed first, then all the modelling and testing were performed to release one version of the project. Each week for each version, the development team met to discuss the progresses or barriers and report the status of the project to the team and project manager, who would decide if further action was required. Every month for Project 1 and every two months for Project 2, workshops for end users were conducted to demonstrate the system and ask for feedback before releasing major developments. Approximately every six months, a big version of the project was released during a workshop for all the stakeholders. Here, the PCS was demonstrated, and feedback and comments were collected to see if the project was headed in the right direction.

4.1.1.5. PCS scoping and modelling. All project scoping was done and documented at the beginning of the project, including defining all the details to prepare for big releases. Here, the product owner and the project manager made decisions on the to-be configuration process, PCS functionalities, PCS outputs, IT architecture, integration with other IT platforms (such as CAD systems) and products to include in the PCS. The starting point for defining the project goals was the stakeholders' requirements.

Stakeholders' requirements were defined by the product owner, who recorded them in various documents; subsequently, the project manager formalised these requirements using use-case diagrams and flowcharts. Requirements were prioritised based on risks in the very beginning of the project when PCS delivery was split into different versions; they were refined during the project only if the project manager requested it. Stakeholders' requirements were translated into goals by the project manager. These requirements were strongly dependent upon the product variety included in the PCS and the configuration process created by the PCS. Thus, a detailed analysis of the product and the configuration process was performed in Projects 1 and 2.

An overview of the variety of products to be incorporated into the PCS was created at the beginning of the projects. Here, the product knowledge analysis was very detailed, complete and elaborate to understand the product structure and variety. Hence, the project managers created large PVMs and discussed all the rules and features of the product. A PVM is a hierarchical representation of a product family structure composed of two parts: one shows the elements (e.g. groups, modules, parts, components) a product can consist of, and the other describes the possible variants and constraints of these elements (Hvam et al., 2008). An excerpt of PVM is presented in Fig. 3. This hierarchical representation allows for the division of the projects following the product hierarchy. A PVM is a representation that is specifically adopted in developing PCS because it enables detailed analysis and visualisation of the product range to be modelled inside the PCS. Beside PVMs, the product knowledge was documented in CRC cards, which are cards that describe characteristics and properties of the PVM elements (classes). A CRC card defines a class by including the class' name, its possible place in a hierarchy, a date, the name of the person responsible for the class, the class' task (responsibility), the class' attributes and methods and which classes it collaborates with (collaboration) (Hvam et al., 2008). In addition, documentation using a class diagram – a type of UML diagram that describes the structure of a system – was done for the cases requiring further representation. The project manager analysed, documented and visualised the configuration process using detailed flowcharts at the beginning of the project. Here, all the PCS functionalities and outputs were defined, such as generating the bill of materials, price calculations, a technical summary and a sales report.

IT architecture for required software developments was determined in the initiation phase and then documented. Further IT developments such as extra features and integrations were planned in the beginning as well based on stakeholders' requirements.

There were also several frequent steering committee meetings to deal with reporting on the project's status and making decisions about the project (e.g. if the project needed more resources, this had to be evaluated and decided upon in the steering committee meetings).

4.1.1.6. Implementation and software integration. The PCS implementation consisted of all the activities required to get the software up and running. Modelling the product process-related knowledge into the PCS and writing related programmes were performed by the configuration engineer. Integrating the PCS with other IT systems (e.g. CAD for 2D and 3D images, ERP for the product-related knowledge, CRM for the knowledge related to the customers, simulation tools to simulate a specific process) were performed by the developer. In this phase, the IT professionals performed all the activities to complete the defined PCS version.

4.1.1.7. Testing. There were two rounds of testing performed every two months for Project 1 and every three months for Project 2. The first round was technical IT team testing. For each PCS project release, the project manager, the configuration engineer and the developer met to define test cases and test the system from a technical perspective, ensuring the system did not have any bugs.

The second round comprised stakeholder testing. Here, domain experts verified the knowledge modelled in the configurator. This round also contained unplanned meetings and workshops with domain experts to gather stakeholders' feedback after each release, such as their opinions about user interface.

Once the PCS version passed both technical and stakeholder testing, it received final approval and entered the deployment phase. For small issues such as minor changes in user interface or adding new knowledge to the product, the PCS version went back to the development phase to be fixed. For some fundamental requests with a high amount of required time and resources, a new project was planned and scoped (e.g. one fundamental change was a new integration that was critical but that was overlooked during the project scoping and planning phases).

4.1.1.8. Deployment. Deployment made employees use the new PCS and abandon the previous PCSs and routines (made of Excel sheets and manual actions). Deployment was done one time per project when the system was totally completed, ready for use and approved. The project manager presented the approved system to the stakeholders as the new system that would be used. The manager also answered their questions regarding system functionality and quality. Any major criticisms could lead to structural changes. For example, they asked to include the product packaging in the PCS models. However, adding packaging in a PCS required a new discipline of knowledge as well as making structural changes to the PCS. In this case, stakeholders had to ask for a new project. In another example, they asked that the packing and transportation (including insurance and prices) be optimised — these requests were assigned as project maintenance and updates.

[1] A release is a project deliverable; it is complete and ready to be delivered to the project customer.

4.1.1.9. Maintenance and updates. In the two RUP-based PCS projects, maintenance consisted of fixing bugs, continuously improving the system after its launch and implementing changes in the product structure or in the configuration process. In these two projects, the IT professional responsible for a part during the development phase was responsible for the same part in the event of necessary maintenance and updates. When the professional was not available or was assigned to other projects, the maintenance task waited until the professional was available. Maintenance was prioritised by domain experts based on the mentioned risks and documented in PVMs, CRC cards, class diagrams and extra documents (e.g. notes and minutes of meetings).

4.1.2. Scrum-based projects

4.1.2.1. Project values. In the two Scrum projects, the main guiding value was rapidity in delivering tangible value.

4.1.2.2. Roles and main tasks. The project roles involved domain experts (six in Project 3, five in Project 4), end users (10–13 in Project 3, 8–11 in Project 4), stakeholders (17–20 in Project 3, 15–17 in Project 4), a product owner, a project manager, a Scrum master, an application manager and the development team (two configuration engineers, two developers and a tester for Project 3 and a configuration engineer, a developer and a tester for Project 4).

In these two PCS projects, all the domain experts were chemical engineers from the sales department with different expertise in chemical process, technical services and sales management. As for RUP projects, these experts were responsible for transferring the product knowledge and system requirements and verifying and validating the knowledge inserted into the PCS. Unlike RUP projects, these experts were selected based on the different disciplines involved in each part of the project. End users, stakeholders, the product owner and the project manager had similar roles as in RUP projects. The application manager was added as a new resource to the project; this person managed the IT team and all of the projects and project managers working in parallel. The Scrum master arranged the daily meetings and maintained a record of them to resolve any production problems and coordinated activities between the Scrum team and the rest of the organisation. Specifically, in these two Scrum-based projects, the product owner, the application manager and the project manager discussed and established the project scope. The application manager and the project manager were responsible for discussing and defining the IT architecture aligned with the development team. The configuration engineers (modellers) modelled all the product knowledge into the platform based on the project manager planning and scoping the project. The developers were responsible for aligning the integration and plugins (i.e. ERP, CAD, CRM or simulation systems). Finally, the tester was responsible for testing and verifying each user story.

4.1.2.3. Task assignment. Scrum projects were flexible in terms of task allocation. The development team members were allowed to work in parallel on different projects, accepted user stories from different projects and even changed roles if needed. Multiple resources were allocated to several projects based on task and priority. This means that not only could the team work across projects and in parallel to save time and increase efficiency but they could accept different tasks based on their expertise and availability. Domain experts were used in parallel when a particular source of knowledge was needed; parallel use of team members and domain experts was also allowed.

4.1.2.4. Activity organisation. Project planning was defined with different levels of detail at different points in time. In the initiation of each project there was an overall description, assessment and planning of the project at a high level of abstraction. Projects were broken down into very small releases to be delivered one by one every three weeks. For each sprint, a detailed planning was done. This planning activity took place regularly every three weeks.

A preliminary activity in Scrum-based projects was backlog grooming. Here, the product owner and the project manager broke the task into smaller items/activities that composed the backlog. A Scrum grooming, where the project manager, project owner and Scrum master groomed the tasks, took place before each planning meeting. Then, the tasks were prioritised based on resource availability.

During sprint planning, the development team selected a subset of product backlog items and agreed to complete them in three weeks. Sprint planning started with a Scrum planning meeting attended by the whole team, the business owner and the project manager. Typical questions this meeting answered were as follows: What is this sprint's capacity for different resources (story points per person)? What tasks from the backlog should be prioritised? What are the estimated story points (as per the project managers) and resources? The Scrum master managed the sprint planning. This person knew the available time of each resource in advance and was capable of planning for this adequately. The team members inserted their availability one week in advance to facilitate the Scrum planning. Team members created sprint backlogs during sprint planning.

After the activities of a sprint were defined, the sprint began. During each sprint, which in these two projects lasted for three weeks, the development team worked to develop the part of the system that had been defined in the sprint planning. Every morning during a sprint, a daily Scrum meeting (15 min) was conducted to manage workflow. In this meeting, the IT team considered the tasks done the day before and the tasks to be done that day and discussed the project obstacles and blockers. Typical questions included, What did you do since the last Scrum meeting? Did you encounter any obstacles? What will you do before the next Scrum meeting?

At the end of each sprint, a sprint review (retrospective) meeting was attended by development team and (most of the time) the product owner. They reviewed what went well in the sprint that had just ended and what could be improved; they then defined the sprint that had to start immediately after the meeting (i.e. task prioritisation) and planned it in detail (i.e. assigned tasks to resources). This review was internal and IT-related; it concerned how to work better and how to make the team more efficient. Typical questions for this meeting were, What went well in this sprint? What went poorly in this sprint? What did you learn? What are the suggested improvements for the next sprint?

Every three weeks, at the end of each sprint, there was a sprint review and a feedback meeting that included the team members, stakeholders and product owner. The stakeholders and Scrum team inspected the delivered product. The lessons learned and improvements from the previous sprint were documented. The questions asked during the sprint review meeting were, What are the testing results for the deliverables from this sprint? What is the feedback regarding the tested materials? What changes (upgrades) are needed?

Every month, a steering committee meeting was attended by the project owner, project manager, domain experts and the stakeholders, including relevant top managers. The purpose of this meeting was to report to the top managers and stakeholders who were not directly involved, observe the project's progress and gather feedback. Here, participants discussed the overall results and whether everything was headed in the right direction.

4.1.2.5. PCS scoping and modelling. The project scoping was done at a high level of abstraction at the beginning of the project and then detailed and refined sprint by sprint. The product owner and the project manager made the same (i.e. the same as during the RUP PCS projects) decisions about the project goals, starting from the stakeholders' requirements as defined by the product owner via documents and notes. The project manager then formalised these requirements with user stories and acceptance criteria. The project manager translated the requirements into goals and prioritised them based on the business value of each sprint. In the two Scrum projects, the knowledge regarding product and configuration processes were documented mainly as user stories.

Product knowledge was analysed using only the readily available documents and notes at the beginning to define the project's overall scope. Then, in each sprint, the detailed product-related information was included in the user stories reported in the backlog without using PVM, CRC cards or class diagrams. In the end, the product knowledge documentation was located in the user stories and in each PCS release. The configuration process was documented in user stories and in the Scrum platform (Jira) during the sprints.

Similarly, the IT architecture for required software developments was managed in the sprints. Further IT developments such as extra features and integrations were planned at the beginning and done during the relevant sprint.

4.1.2.6. Implementation and software integration. The nature of the PCS implementation activities did not differ from the PCS RUP-based projects. In the Scrum projects, these activities where performed in each sprint according to the sprint planning in order to complete the defined release.

4.1.2.7. Testing. Testing was performed per sprint every three weeks, including both technical and stakeholder testing. The tester, who was part of the IT development team, performed this activity by using the test cases that the IT team had predefined and planned. Technical testing of each user story was done during the sprint. Before the end of the sprint, the release was sent to stakeholders for testing. Therefore, the team members received feedback from users every three weeks, and when possible, from the product owner and the steering committee.

In the two Scrum-based projects, requests for changes were all within the scope of the projects and immediately defined and inserted as new user stories in the backlog.

Finally, when a version passed the test, the tester and the stakeholders approved the executed sprint, making it possible to arrange the next sprint with upcoming tasks. Otherwise, based on feedback or comments on knowledge validity, new tasks were defined for the next sprint — some of the modelling and development tasks has to be repeated based on a request or on knowledge that had to be corrected.

4.1.2.8. Deployment. Like in the RUP-based PCS projects, system deployment was done once per project — when the system was totally completed, ready to use and approved. While in theory opposed to the Scrum principles, this approach has been adopted because of the high amount of interdependencies presented in the considered PCSs. Indeed, it is very complicated to proceed to the deployment of a partial PCS because this leads to incomplete or even inconsistent (thus unbuildable) product configurations. The insertion of a complete set of constraints (which imply touching all the dimensions of the knowledge to be inserted in the PCS) is necessary to be able to perform a complete and feasible configuration even for a subset of the considered products. Once all of the product feature interdependencies are considered, the system is useable in a live environment. So while each release is validated with users and stakeholders at the end of each sprint, as typically done in Scrum based software development projects, the actual deployment is not done in a phased or staged approach but rather at once with a consistent system at the end of the project. Deployment is thus done on the basis of the final release for all the involved and potential stakeholders. The product owner took responsibility for changing the routines and people's mind-sets in order to replace the old systems in use. In the two projects, the approved PCSs were adopted immediately, as they were approved or modified by involving stakeholders in frequent meetings. This frequent involvement ensured the constant engagement of stakeholders during the project and the PCS' good alignment with their expectations.

4.1.2.9. Maintenance and updates. In these two projects, maintenance was assigned to any one member of the development team as soon as possible by including the request in the sprint. Unfortunately, in these two projects, maintenance suffered from little documentation, as it was based mainly on user stories.

4.2. How PCS project challenges were addressed

In order to gather the effects of the PCS challenges from Scrum in comparison with RUP, factual data (collected mainly though archival data and interviews) and judgements provided by PCS development team members (collected through questionnaire-based semi-structured interviews) are provided. Table 3 presents the factual data that provides further elements characterising RUP- and Scrum-based PCS projects on one side and PCS challenges for each project on the other. Table 4 presents the results of the interviews on the influences of Scrum compared to those of RUP on PCS challenges. While Table 5 provides objective evidence about whether and how a certain challenge was faced, Table 6 provides judgement-based evidence and explanations of why Scrum performed better or worse than RUP in facing PCS challenges.

In Table 3, the first column refers to the PCS projects challenges as reported from literature and highlighted in Section 2.3 and Table 1. The second column addresses the details of each of the challenges based on the description and details of each of the challenges highlighted in Table 1. The rest of the table provides the numbers and explanation to compare the performance of the RUP and Scrum projects.

Tables 2 and 3 suggest that Scrum-based PCS development projects are faster than RUP-based ones in terms of total duration (10–12 months versus 15–20 months) because they involve more domain experts (5–6 versus 2–3) and use more of them in parallel (2 versus 1), use more IT professionals (3–5 versus 2) and use more domain experts and IT professionals in parallel (3–5 versus 2). However, RUP projects use almost the same IT resources (9–14 pm versus 10–14 pm) and more domain expert resources (5–7 pm versus 3–4 pm). In addition, RUP-based projects are more behind the schedule to keep up with product changes than Scrum-based ones (1–2 pm versus 1–2 pm). Finally, the consumption of resources for maintenance/updates are higher for RUP versus Scrum (1–3 pm versus 1 pm).

As far as the product-related and knowledge acquisition challenges, RUP-based PCS projects used tools (like PVM and class diagrams) to handle the product range complexity, whereas Scrum-based ones did not use tools to visualise and gain an overview of this product variety-related complexity. Regarding knowledge acquisition challenges, RUP-based projects presented more errors (approximately 10–13 versus 2–4) and more rewritten rules (approximately 15–30 versus approximately 3–5) due to a low-quality acquisition of knowledge. However, while in Scrum the lower percentages of rewritten rules ($< 0.4\%$) are due to multiple and frequent validations, in RUP the limited percentages of rewritten rules ($\leq 2.5\%$) are due to accurate and systematic

Table 2
Main project figures.

Comparison aspect	RUP-guided PCS development		Scrum-guided PCS development	
PCS projects	*Project 1 (RUP)*	*Project 2 (RUP)*	*Project 3 (Scrum)*	*Project 4 (Scrum)*
Total duration (months)	15	20	12	10
Net duration (months)	8	12	11	8
Project split	2 iterations	3 iterations	10 sprints	9 sprints
Number of attributes	∼1500	∼1900	∼2000	∼1500
Number of constraints	∼1000	∼1200	∼1400	∼1000

knowledge elicitation and formalisation. Finally, they were slower in fulfilling requests to update PCS knowledge (1–2 months versus 1–2 weeks).

As far as the IT (technical) challenges, Scrum-based PCS developments encountered fewer serious IT challenges that, when encountered, were solved more rapidly (1–2 weeks versus 1–2 months) due to the presence of more experts who were specialised in different fields. In Scrum-based PCS projects, users were more satisfied because they tested the system in every sprint and could see the deliverables in the system. Further, the user interfaces were more aligned with user requirements due to their iterative approval. RUP-based projects had less frequent communication with stakeholders (every 1–2 and 2–3 months versus every 3 weeks). Finally, regarding organisational challenges, Scrum-based projects did not present episodes of serious resistance to changes, whereas RUP-based projects faced 1–2 episodes.

Table 4 presents the results of the interviews. It reflects the format (with the difference that column 1 was not provided and columns 8 and 9 where collapsed and headed by "Any comment?") and reports exactly the interviews statements as emailed and discussed during interview sessions. The first column lists the main categories of challenges related to the PCS projects reported from literature and highlighted in Section 2.3 and Table 1. The second column presents the statements while the column 3–7 are the Likert-scaled answers provided for the interviewees. Column 8 and 9 provide the interviewees with the opportunity to explain the reasons for their answers to the Likert-scaled statement respectively for RUP and Scrum.

Table 4 shows that on 9 out of 14 of the challenges, Scrum provides better support than RUP. Interestingly, in two aspects (9 — knowledge documentation, and 11 — knowledge visualisation and representation), RUP performs better than Scrum. In two aspects (3 — managing the complexity of the product range, and 8 — knowledge acquisition), RUP and Scrum are almost equivalent. These results indicate that Scrum is generally superior to RUP in facing PCS challenges; however, it presents some limitations on knowledge management, where RUP performs better. However, awareness of the limitation of Scrum in terms of PCS project management would assist practitioners with risk management.

5. Discussion

5.1. SCRUM versus RUP in PCS project activities

Table 5 comparatively synthesises and visualises the results reported in Section 4.1 and some information on PCS development project performance reported in Table 3. Table 5 is designed to highlight the differences between RUP-based and Scrum-based PCS development activities (and performance) and to compare these differences to those observed in RUP-based and Scrum-based general software development activities (and performance). In order to reach the intended comparison objectives effectively, we grouped the first four coding categories reported in Sections 4.1.1 and 4.1.2 (namely project values, roles and main tasks, task management and activity organisation) into a common super-category called organisational approach. Vice versa we split the PCS scoping and modelling category into four subcategories namely stakeholder requirements, product knowledge, configuration process, and goals. Given that this table could be read fast by the audiences without the need to read the entire paper. In order to avoid misinterpretation we replace the technical expression "implementation and software integration" with a more intuitive term "working on the software". Table 5 starts with the organisational approach (something that is usually decided before a specific PCS project starts but that highly influences how the PCS project is conducted), continues with the project realisation activities (that revisit the phases of PCS development procedure which has been proposed by Hvam et al. (2008) considering various methods, including RUP but not Scrum), and ends with the PCS project performance that can be fully appreciated only after the project completion. Table 5 is detailed enough to avoid becoming generic; it is also self-explanatory to allow the present discussion to be limited to the main findings that emerged.

RUP-based PCS projects follow a systematic approach that performs a deep analysis of what has to be developed before starting its realisation; they also document everything very accurately to ensure good implementation and future maintenance and enhancements. Team member assignment is based on specific knowledge and competencies (remember that in PCS projects, the required knowledge and competencies are very differentiated) and tends to follow a sequential approach: if one activity A needs input from activity B, then activity B must be well completed before starting activity A. All the activities must be done in a planned way by specialised people with limited uncertainty. It is important to reduce uncertainty from the beginning of the project to increase quality and efficiency. Scrum-based PCS projects work with a completely different view, as they aim at delivering fast value. This is important for acceptance of the PCS since people can appreciate and become accustomed to the new system. To be rapid, in Scrum-based projects, more IT professionals and even more domain experts are involved and work much more in parallel both within and across roles. To assure correspondence with users' requirements, in Scrum-based PCS projects, there are not only more frequent interactions with stakeholders to gain detailed input but also more frequent testing. Both user need determination and PCS artefact definition are spread over time and evolve. In Scrum-based PCS projects, the team accepts that they will not have detailed control of the overall project, but they do not accept being late with what is planned for the next three weeks. In contrast, in RUP-based PCS projects, teams do not accept moving on without having removed even small uncertainties that may affect the quality and the performance of the project. In order to do this, the teams accept that they can spend more time than planned in the analysis and documentation of products, configuration processes and determining which PCS characteristics to develop. These fundamental organisational differences between Scrum-based and RUP-based PCS projects are not so different from what has been observed in general software projects (Collaris and Dekker, 2010; Noordeloos et al., 2012).

Scrum-based PCS development is faster than RUP-based development. This is due both to the greater concurrent performing of

Table 3
Project performance data.

PCS challenge	Dimension of impact	*RUP*		*Scrum*	
		Project 1	*Project 2*	*Project 3*	*Project 4*
Human resources constraints	1. (a) Time of IT project development team assigned in each project	∼15 months	∼20 months	∼12 months	∼10 months
	1. (b) Amount of use of IT project development team (full time equivalent in person–month [pm])	Conf. engineer: 8 pm Developer: 1 pm	Conf. engineer: 12 pm Developer: 2 pm	Conf. engineers: 11 pm Developers: 2 pm Tester: 1 pm	Conf. engineer: 8 pm Developer: 1 pm Tester: 1 pm
	2. (a) Time of domain experts assigned in each project	2 resources: R1: 8 months R2: 2 months	3 resources: R1: 9 months R2: 2 months R3: 1 months	6 resources: R1: 4 months R2, R3, R4, R5, R6: 1 month for each of them	5 resources: R1: 2 months R2, R3, R4, R5: 1 month for each of them
	2. (b) Amount of use of domain experts (in person–month [pm])	2 resources (1 resource assigned per time): R1: 4 pm R2: 1 pm	3 resources (1 resource assigned per time): R1: 5 pm R2: 1 pm R3: 1 pm	6 resources (1–2 resources assigned per time): R1: 2 pm R2: 1 pm R3, R4, R5, R6: 1 pm in total	5 resources (2 resources assigned per time): R1: 1 pm R2: 1 pm R3, R4, R5: 1 pm in total
Product-related challenges	3. Tools to handle complexity of the product range in PCS project development	PVM helps to handle the complexity	PVM and class diagrams help to handle the complexity	No tool for visualisation and overview	
	4. (a) Management of product changes and time devoted to this	As the project development and maintenance were longer, they were normally behind the schedule to keep up with the changes:		They immediately made a user story for the changes, and one of the resources took over the task based on availability:	
		∼1 month	∼2 months	2 weeks	1 week
	4. (b) Average time per year for maintenance/updates	3 months: around 1–2 person-months	5 months: around 2–3 person-months	2 months: ∼ 1 person–month	2 months: ∼ 1 person–month
IT (technical) challenges	5. Main IT technical challenges and time devoted to solving serious challenges	Integration, IT project management, lack of resource from domain experts		There were few challenges because there were so many different experts in the team who specialised in different fields. Thus, in case of need, the tasks were easily assigned.	
		2 months: ∼1 person–month	1 month: ∼2 person–week	2 weeks: ∼4 person–day	1 week: ∼2 person–day
	6. User assessment of PCS friendliness	They did not complain but were not completely satisfied.		They were satisfied because they tested the system in every sprint and could see the deliverables in the system.	
	7. User interface, requests for user interface changes (if any) and rounds devoted to addressing them	Because the same software was used, the user interface was the same; what was different was how they modelled the product and determined the project scope and details.			
		Good Request for modifications:		Aligned closely with requirements and approved iteratively	
		2 rounds of changes	5 rounds of changes		
Knowledge acquisition challenges	11. (a) Number of errors related to low-quality knowledge acquisition	∼10	∼13	∼4	∼2
	8. (b) Number of rules that were rewritten due to poor knowledge acquisition	∼15	∼30	∼5	∼3
	9. Number of pages (or equivalent of pages) in the project documentation	Complete PVM:		The documentation consisted entirely of the user stories and explanation of the tasks	
		10 pages	6 pages		
	10. Average time between a request to update PCS knowledge and the updating of this knowledge	1 month	2 months	2 weeks	1 week

(*continued on next page*)

activities and the lower amount of activities that must be redone due to erroneous knowledge. They are more aligned with users' requests and highly accepted by users. This is due to the greater stakeholder involvement and more frequent and focused testing. It also leads to superior development efficiency, as witnessed by lower consumption of working time from both IT professionals and domain experts. This is due to more confined chunks of work that reduce complexity, more detailed control of everyday problems that prevent blockers, and easier access to the specific knowledge that is required. Likely, greater collaboration and the

Table 3 (*continued*).

PCS challenge	Dimension of impact	*RUP*		*Scrum*	
		Project 1	*Project 2*	*Project 3*	*Project 4*
Product modelling (knowledge representation) challenges	11. Modalities of knowledge representation and visualisation	PVM		Nothing	
	12. Modalities used to communicate with PCS stakeholders	The communication was available every:		Very high level: daily internal meetings, workshops and testing (every 3 weeks)	
		1–2 months.	2–3 months.		
Organisational challenges	13. Number of episodes of serious resistance to changes	1	2	Never	

Table 4
Results of interviews related to the benefits of Scrum compared to RUP based on PCS challenges.

Challenges related to the management of PCSs	Statements	Strongly disagree	Disagree	Neutral	Agree	Strongly agree	Comments on RUP (why?)	Comments on Scrum (why?)
Resource constraints	1. Scrum reduced the time and resources needed from the IT project development team for PCS projects when compared to RUP.			1	3		• Highly structured overview and framework • Detailed planning at the beginning • Component-based development • Management of software changes	• Faster development in Scrum • More convenient to allocate resources in Scrum • A good estimation tool (user stories and story points) in Scrum
	2. Scrum reduced the time and resources needed from the domain experts for PCS projects when compared to RUP.				2	2	• Iterative testing, validation and improvements from domain experts	• User stories in Scrum helped the business see blockers • Better communication • Frequent constant validation and testing
Product-related challenges	3. Scrum managed the complexity of the product range in PCS project development better than RUP.		2	1	1		• Flexible integration of dedicated tools for product modelling • Product model visualisation • Detailed discussion with stakeholders regarding product-related challenges	• Scrum did not support the visualisation and analysis of product knowledge • Scrum did not suggest a tool for product overview like RUP did • Breaking down the tasks into user stories in Scrum clarified the problem domain
	4. Scrum mitigated the challenges of product updates and the development rate in PCS projects when compared to RUP.				3	1	• Slow development and maintenance procedure • Full understanding of the product structure at the beginning of the project to create a full picture	• Faster maintenance in Scrum • Faster development in Scrum • User stories in Scrum hastened all the phases of development

(*continued on next page*)

possibility of substituting written memory with human memory played an important role in reaching these results. Overall, these differences in performance between Scrum-based and RUP-based PCS development are similar to those observed in general software development (Collaris and Dekker, 2010; Noordeloos et al., 2012).

Table 4 (*continued*).

Challenges related to the management of PCSs	Statements	Strongly disagree	Disagree	Neutral	Agree	Strongly agree	Comments on RUP (why?)	Comments on Scrum (why?)
IT (technical) challenges	5. Scrum mitigated the technical software challenges in PCS projects when compared to RUP.				4		• Slow and detailed development	• The IT technical challenges were quickly identified and prioritised in Scrum. • Allocation of tasks to the appropriate resources in Scrum
	6. Scrum mitigated the challenges of design thinking, input/output requirements and system friendliness in PCS projects when compared to RUP.				2	2	• Tools for functional and non-functional requirements	• Scrum employed user stories and sprint reviews for requirement analysis. • Good communication in Scrum • More iterative testing and feedback in Scrum
	7. Scrum improved the user interface in PCS projects when compared to RUP.				3	1	• Iterative testing, but no fixed meeting	• Fixed sprint reviews for discussions in Scrum • Scrum enabled responsiveness to rapid changes in stakeholders' requirements
Knowledge acquisition challenges	8. Scrum performed better regarding PCS project knowledge acquisition when compared to RUP.			3	1		• Framework and tools for knowledge management • Comprehensive knowledge acquisition at the beginning of the project	• Tacit knowledge in Scrum • Poor documentation in Scrum • No structured framework or tool for knowledge acquisition in Scrum
	9. Scrum performed better regarding PCS project knowledge documentation when compared to RUP.	1	3				• Comprehensive documentation of knowledge	• Once the sprint was done, all the documented knowledge from the user stories was difficult to find and document in Scrum • Very light and agile documentation in Scrum
	10. Scrum performed better in terms of maintaining and updating the knowledge in the maintenance phase of PCS projects when compared to RUP.				4		• Explicit available knowledge	• Scrum successfully kept track of user stories and bugs. • Fast maintenance in Scrum due to quick adaptation to required changes

(*continued on next page*)

Most of the differences detected between Scrum-based and RUP-based PCS development are similar to the differences observed for general software development. However, one difference is highlighted for PCS — namely, product knowledge documentation. While in Scrum, user stories are used both in PCS and general software development, in RUP, tools such as PVM, CRC cards and class diagrams are used only in PCS projects. This may seem like a small difference, but it is significant because product knowledge elicitation, documentation and updates determine the quality of a PCS to such an extent that this can cause a team to abandon a PCS. Even the first case reported in the literature about a PCS showed heaviness of PCS update and maintenance due to changes in product characteristics (Barker et al., 1989). Knowledge acquisition and product modelling (knowledge representation) are among the most important challenges of PCS projects (Kristjansdottir et al., 2018). So, the Scrum limitation on knowledge representation should not be underestimated, even though the Scrum-based PCS projects analysed here (as well as the other Scrum-based PCS projects performed in the case company) were successful.

Table 4 (*continued*).

Challenges related to the management of PCSs	Statements	Strongly disagree	Disagree	Neutral	Agree	Strongly agree	Comments on RUP (why?)	Comments on Scrum (why?)
	11. Scrum performed better regarding PCS project knowledge visualisation and representation when compared to RUP.		2	2			• Visualisation tools such as product tree structures	• No product visualisation due to the speed of development in Scrum
Product modelling (knowledge representation) challenges	12. Scrum performed better in terms of communicating with stakeholders regarding the product model in PCS projects when compared to RUP.				2	2	• No fixed meeting with stakeholders • Iterative testing	• Fixed sprint review meetings in Scrum helped a great deal. • Satisfactory relationship with the customer in all sprints in Scrum • Continuous iterative testing in Scrum
Organisational challenges	13. Scrum performed better in terms of supporting change management regarding resistance to using the PCS when compared to RUP.				1	3	• Not very successful in terms of change management	• Alignment of the requirements in Scrum • Scrum was fast enough to keep up with the changes. • Scrum was very good in terms of involving the stakeholders.
	14. Scrum performs better in terms of supporting the agreements of PCS projects' scoping when compared to RUP.					4	• Implementation difficulties due to the lack of communication • Failure in change management	• Breaking down the tasks helped a great deal in scoping the project in Scrum • Acceptance criteria for each user story helped team members to understand the stakeholders and the scope in Scrum

One similarity observed between Scrum-based and RUP-based PCS development which is different from what is reported in the literature for general software development projects. In the considered PCS development projects, the deployment has been done once the full system was ready. In order to have a PCS that assures feasible configurations even for a subset of products, all relevant constraints had to be inserted. In addition, to unlock the benefits of PCS in the considered cases, a complete family of products was needed to be inserted and implemented. However, we suggest that this finding likely depends on the complexity of the product and on the way the configuration task is organised before the introduction of PCS. Therefore, no doubt that this specific finding deserves further enquiry.

5.2. Scrum versus RUP in facing PCS project challenges

Table 6 summarises the results presented in Section 4.2. It concerns the difference between Scrum and RUP in overcoming PCS challenges. The importance of PCS challenges is taken from Kristjansdottir et al. (2018). Here, Scrum performs better than RUP in facing more PCS challenges, but not for all of them. Hereafter, these results are discussed by considering how the key differences of RUP and Scrum-based PCS projects reflect PCS challenges.

5.2.1. Knowledge documentation

The RUP-based PCS projects strongly relied on configuration processes and product documentation using PVM and CRC cards. The latter performed well for knowledge modelling and visualisation, were used as a basis for discussions during meetings and could easily be integrated into the RUP process. Scrum avoided the need for heavy documentation: knowledge representation and product modelling were performed by modelling/expressing all the product specifications and constraints in the form of user stories. This had the advantage of driving the scope of iterations by user story selection, enabling faster development (Wautelet et al., 2014). Nevertheless, employing user stories also made it impossible to visualise knowledge and to express the full complexity of a PCS. In terms of traceability, user stories did not allow consistent, long-term documentation of developments (Dwivedi, 2013). Furthermore, many constraints could not be expressed in natural language in the form of user stories and therefore required more formal documentation. As a result, in relation to some specific PCS challenges where knowledge documentation played an important role in product modelling and representation and some aspects of knowledge acquisition, maintenance and update (e.g. Aldanondo et al., 2000; Ardissono et al., 2003; Felfernig et al., 2000; Haug and Hvam, 2007; Heiskala et al., 2007), Scrum performed weakly, or at least was not better than RUP.

5.2.2. Knowledge elicitation and validation

In RUP-based projects knowledge, is elicited and systematised at the beginning of the project through specific representation tools (e.g. PVM, CRC cards and class diagrams). These representation tools allow for a better understanding of product knowledge

Table 5
RUP versus Scrum in PCS developments and comparison with RUP–Scrum differences in general software development.

	PCS projects		Different from general SW?
	RUP-guided PCS development	Scrum-guided PCS development	
Organisational approach	• Progress systematically reducing risks • Make best use of specialisations sequentially • Very detailed analysis and planning at the beginning • Release the PCS in big chunks • Moderate contact with stakeholders during the project	• Progress rapidly delivering value • Use more IT professionals and more domain experts • Use them in parallel and interchangeably • Plan, develop, test and deliver PCS parts such that they are small enough to be completed in three weeks • Fast, high-level planning at the beginning, focused detailed planning every three weeks and face-to-face daily microplanning and control	No
Stakeholder requirements	Defined by product owner, who records them in any kind of document; subsequently formalised by the project manager		No
	• Using use-case diagrams and flowcharts • Prioritised (based on risks) in the very beginning of the project when PCS delivery is split into different versions • Refined if requested by project manager	• Using user stories and acceptance criteria • Prioritised (based on business value) in different sprints • Detailed and refined sprint by sprint	
Product knowledge	• Much detailed at the beginning • Making big PVMs and discussing all the rules and features of the product • Analysed and visualised through PVM, documented in PVMs and CRC cards and eventually in class diagrams	• Overall analysis at the beginning to define the project's scope • Detailed analysis in each sprint using user stories reported in the backlog • Not visualised	Yes
Configuration process	• Analysed, documented and visualised using flowcharts in a detailed way at the beginning of the project	• Documented in user stories and in Jira (the Scrum platform) in different sprints along the project • Not visualised	No
Goals	Stakeholder requirements are translated into goals by the project manager. The decisions on the to-be configuration process, PCS functionalities, PCS outputs, IT architecture, integration with other IT platforms (such as CAD systems), products to be included in PCS etc. (i.e. scoping decisions) are made by the product owner and project manager		No
	• In a very detailed way at the beginning of the project • Carefully documented using also PVM and CRC cards, class diagrams and visualised in PVMs	• At overall level at project beginning, specifying acceptance criteria, PCS features, user interfaces, stakeholder requirements, extra IT development, etc. • More detailed decisions made every three weeks in the middle of each sprint for the subsequent one	
Working on the software	Modelling product process-related knowledge into PCS, writing programmes and realising integrations of PCS with other IT systems (such as CAD, ERP, CRM, simulation tools). The activities are not different between RUP and Scrum, even though they are done with a different organisation and timeline (see first row)		No

(*continued on next page*)

between domain experts and IT professionals and are a first control and validation of the knowledge by domain experts. At the end of the version development, domain experts test and validate the knowledge and the system though the inspection of the PCS artefact version. In Scrum-based PCS development, knowledge is incrementally elicited from domain experts, immediately implemented in the PCS and validated by domain experts by inspecting the PCS artefact in each sprint. Consequently, in RUP, the quality of the whole knowledge encompassed in a PCS is guaranteed by checking all the knowledge before starting the project (because it has been incorporated into the PVM) and after each release, while in Scrum, small amounts of knowledge are elicited, controlled and inserted directly into the PCS and test per sprint. Therefore, RUP performs better than Scrum in managing the complexity of the product range as well as knowledge visualisation and representation in PCS project development, which are recognised in several works (e.g. Aldanondo et al., 2000; Ardissono et al., 2003; Felfernig et al., 2000; Forza and Salvador, 2002b,a; Haug and Hvam, 2007; Heiskala et al., 2007) to be specific aspects of product-related challenges and product modelling challenges, respectively.

5.2.3. Collaboration, support and overall momentum

Scrum-based projects involve more IT and more domain experts in total and in parallel. This is a sign that the project is vital for the organisation and that enough resources are provided. More people formally assigned in the projects means more responsibility for results and immediate support, when needed. In

Table 5 (*continued*).

	PCS projects		Different from general SW?
	RUP-guided PCS development	Scrum-guided PCS development	
Testing	Technical IT team testing (around every 2–3 months) • Several frequent IT team meetings Stakeholder testing (around every 2–3 months) • Some un-planned meetings and workshops to gather stakeholder feedback after each release • Sometimes feedback arrive too late, and fundamental changes to the project are needed • If the PCS version passes both technical and stakeholder testing, it receives the final approval and passes to deployment. • Otherwise, (a) for small issues, it goes back to the development phase to be fixed, while (b) for fundamental requests with more than a specific amount of required time and resources, a new project will be planned and scoped.	Technical testing (during sprint for each user story), frequent testing by software specialist tester based on the test cases predefined and planned by the IT team Stakeholder testing (e.g. around every three weeks) • Team members, stakeholders and the product owner inspect and discuss the release in the sprint review and feedback meeting (every three weeks). • If the test of the prioritised user story is passed, then the tester and the stakeholders will approve that sprint and move on to the next user stories. • If changes and further developments of the PCS are needed, than they are immediately defined and listed in the backlog as a new user story.	No
Deployment	• Deployment is done once per project when the system is totally completed, ready to be used and approved • Make employees using the new PCS and abandon the previous system made of Excel sheets and manual actions		Yes
	• The project manager presents the approved system to the stakeholders as the new system to be used and answers their questions • If stakeholders need something else, they have to ask for maintenance or even a new project	• Product owner takes responsibility for changing the routines and mind-set to replace the old system in use • The approved system is expected to be used immediately since this is facilitated by frequent meetings and constant engagement of stakeholders during the project	
Maintenance and updates	• Wait for the availability of the developer of the part to be updated • Maintenance is advantaged by good documentation in PVMs, class diagrams and extra documents such as notes	• Are assigned to any one of the PCS developers' team • Maintenance suffers from little documentation that is mainly user stories	No
Performance	• Slower, more consumption of resources, lower user satisfaction and acceptance, more need for maintenance, slow and behind the schedule maintenance and updates	• Faster, more use of resources, higher user satisfaction and acceptance, less need for maintenance and fast maintenance and updates	No

Scrum, the projects progress rapidly, valuable outputs become visible to stakeholders much earlier and new valuable outputs are continuously delivered. This creates momentum in the organisation, and this momentum helps to find support in a virtuous cycle. In addition, this process both presents small bites of changes at a time, thus making them more digestible, and offers opportunities to provide feedback. This feedback is immediately considered and greatly affects the project. Altogether, this method of working reduces resistance to change. The final consequence is that the main PCS challenges (i.e. organisational challenges) (e.g. Ariano and Dagnino, 1996; Barker et al., 1989; Forza and Salvador, 2002a; Heiskala et al., 2007) are better faced with Scrum.

5.2.4. Consumption of resources

Scrum-based projects consume fewer human resources than RUP-based ones even though they involve more IT specialists and more domain experts. This is not due to only the reduction in the documentation overload, but it is based on several interrelated aspects of the Scrum way to work. There are other inefficiencies removed via the Scrum approach that deserve ad hoc investigations. Overall, the resource constraint challenges (e.g. Aldanondo et al., 2000; Ariano and Dagnino, 1996; Barker et al., 1989; Forza and Salvador, 2002b; Heiskala et al., 2005) benefit from Scrum since it reduces the consumption of resources in PCS projects.

5.3. Threats to validity

This section exposes the threats to the study's validity and recalls the main adopted countermeasures. The discussion is organised via three aspects: construct validity, internal validity and external validity.

With respect to *construct validity*, which 'is the extent to which we establish correct operational measures for the concepts being studied' (Voss et al., 2002, p. 211) and 'reflects to what extent the operational measures that are studied really represent what the researcher have in mind and what is investigated according to the research questions' (Runeson and Höst, 2009, p. 153), the main threat is that the interview questions are not interpreted by the interviewees in the way the interviewers intend. To increase construct validity, three forms of triangulation were implemented: *observer triangulation*, *data (source) triangulation* and *methodological triangulation* (Runeson and Höst,

Table 6
Summary of RUP and Scrum contributions in facing PCS project development challenges.

PCS challenge	Importance of the PCS challenge	More effective method to face the PCS challenge	Key justifications
Organisational challenges	Very high	Scrum	Scrum performed better regarding communication, involving stakeholders and their requirements and change management.
Knowledge acquisition challenges	High	RUP	RUP performed better because of the available tool for knowledge management, comprehensive knowledge acquisition and comprehensive documentation.
Product modelling (knowledge representation) challenges	Medium	It depended on the aspect being considered	RUP empowered the team with knowledge visualisation and validation tools. Scrum empowered the team with strong communication, meetings and stakeholder involvement.
IT (technical) challenges	Medium	Scrum	Scrum performed better due to superior task prioritisation and allocation, iterative testing and continuous feedback.
Resource constraints	Low	Scrum	Scrum performed better because of the fast development and iterative releases, management of the impediments and flexibility in changing roles.
Product-related challenges	Low	It depended on the aspect being considered	RUP empowered the team with knowledge visualisation and modelling tools for a full understanding of the product structure. Scrum empowered the team with strong collaboration, frequent knowledge validation and fast development, maintenance and updates.

2009). First, the research team that performed the interviews and document analysis consisted of three researchers, allowing *observer triangulation*. Interviews were recorded (with the authorisation of the interviewees) and subsequently systematically analysed by researchers both independently and as a team to avoid misinterpretations. Results analysis was conducted with the entire research team to avoid misinterpretations. Regarding *data (source) triangulation*, the same questions were asked of all four interviewees, thus ensuring multiple response sources. Several interviews were conducted using the same set of questions with people playing (or having played) various roles in the PCS development team. By collecting data on the same aspects from different points of view, the researchers increased the validity of the data used to make inferences. In addition, both *data (source)* and *methodological triangulations* were applied by using factual data related to each PCS challenge in addition to qualitative interviews, non-systematic direct observation and company documents produced during each project (which the researchers had completely at their disposal). The long-term trusting relationship between the researchers and the investigated company provided multiple advantages for increasing construct validity (Runeson et al., 2012), such as a better understanding of how participants interpreted terms, more information transparency and correctness and the participants spending more time providing relevant information.

Internal validity 'is of concern when causal relations are examined' (Runeson and Höst, 2009, p. 154) and 'is the extent to which we can establish a causal relationship, whereby certain conditions are shown to lead to other conditions, as distinguished from spurious relationships' (Voss et al., 2002, p. 211). The major threat to internal validity is that interview answers regarding the impact of RUP and Scrum on PCS development challenges might reflect respondents' biased opinions on the adopted development method rather than how the method truly performs. Also, the roles played by the respondents within the PCS development team might influence their knowledge of the project management and engineering activities being performed. Consequently, the knowledgeability of interviewees in answering some of the questions may vary. However, all the countermeasures the present researchers adopted for construct validity allowed them to increase internal validity — in particular, the justifications provided by each interviewee about each one of his or her judgements (see columns 8 and 9, Table 4) and confronting their judgements (Table 4) with objective data (Table 3). In addition, using multiple cases (with both theoretical and literal replications) permitted the researchers to detect a similar pattern of causal relations between RUP and Scrum and PCS challenges across cases, thus increasing the validity of the findings. The other threat to internal validity is the duration of RUP and Scrum at the case company. Based on Fig. 1, the duration of working and experiencing RUP is 5 years while this period is around 3 years for the Scrum projects. Even though the durations of experiencing RUP and Scrum are long enough for this research, we signal that the duration of experiencing RUP is longer than the duration of utilising Scrum for PCS projects at the case company.

The threat to *external validity* concerns the fact that the results may not be generalisable beyond the investigated case studied (Runeson and Höst, 2009; Voss et al., 2002). Even though the findings of this study are based on multiple projects, they are drawn from a single organisation. The results can be extended and are likewise relevant to companies that have similar characteristics to the company analysed here that are planning to begin an agile transformation in comparable circumstances. In addition, several, if not all, of the mechanisms presented here to explain the impact of RUP and Scrum on project performance and PCS challenges are valid far beyond the kind of company that was analysed. However, further studies should be performed in different contexts, such as with companies that are moving to Scrum without having used RUP before or in small and medium enterprises where some IT experts are external to the company, and the internal domain experts are overloaded. The authors of this article expect that the results hold globally, even in these contexts; however, they also expect some peculiarities that could call for specific adaptations of Scrum and RUP for PCS projects.

6. Conclusions

PCSs can bring substantial benefits to companies, such as shorter lead times for generating quotations, fewer errors, use of fewer resources, less routine work and improved on-time delivery (Haug et al., 2019; Hvam et al., 2013; Trentin et al., 2012, 2011). Unfortunately, PCS projects present certain difficulties, such as resource constraints, product-related challenges, technical challenges, knowledge acquisition challenges, product modelling challenges and organisational challenges (Kristjansdottir et al., 2018). The method used for planning, developing,

implementing and maintaining PCS projects may influence these challenges. Agile methods like Scrum are renowned for overcoming certain shortcomings of the traditional methods of software development. Unfortunately, the differences between PCSs and other software development projects concern several aspects — knowledge complexity and extensions, maintenance and documentation, knowledge modelling techniques, etc. (Shafiee et al., 2018). PCS projects being different from other software development projects, it is not obvious that the advantages of Scrum compared to those of the RUP hold for PCS projects as well. For these practically and scientifically relevant reasons, the present paper investigated the differences between Scrum and RUP in managing PCS projects as well as the consequent differences between their impacts on the challenges of PCS projects.

The study provided empirical evidence that both RUP and Scrum can be effective to develop PCS. However, the way the work proceeds differs vastly in the two approaches. In Scrum, user stories for PCS specifications, frequent interaction with users and domain experts and focusing sprints on the most valuable aspects make PCS development more accepted and supported by the organisation. This is similar to what happens with other software engineering projects. However, the speed and flexibility in specification definition and in knowledge formalisation are obtained at the expense of knowledge visualisation, as PVM and CRC cards are not replaced by equally capable tools for knowledge representation. So, for one of the aspects that most differentiates PCS projects from other software engineering projects, Scrum presents a weakness or at least is not able to perform better than RUP.

The comparison of RUP- and Scrum-managed PCS projects highlighted that both methods helped with PCS challenges; however, they did so differently. Scrum performed better in facing organisational challenges – recognised as often being the biggest challenges – and also in facing IT challenges and resource-constraint challenges. In contrast, RUP performed better in knowledge acquisition challenges. So, even though Scrum presents several advantages over RUP, it is not automatically the best choice.

The results of this study can be used as guidelines and warnings for transitions to Scrum in companies conducting PCS projects. In particular, such companies should focus on finding the right amount of PCS requirements and design representation either in user stories or via an adequate way to integrate artefacts, like PVM and CRC cards, keeping these fully consistent with the knowledge depicted in the user stories.

Overall, these results show that Scrum has several advantages but is lacking in the areas of documentation and visualisation of PCS knowledge. Notably, knowledge management is crucial for PCS, and once again, the results confirm that PCSs have special needs in terms of knowledge management that differentiate them from other IT systems (Shafiee et al., 2018). Therefore, both practice and research should consider integrating adequate documentation tools and knowledge representation in managing PCS projects. Several solutions can be envisaged in order to bypass these shortcomings. The most promising possibility would be to integrate PVM and CRC cards into Scrum, even though the process is driven by user stories. Both requirement–representation artefacts could be used in parallel, but consistency should then be ensured. A computer-aided software engineering tool can be specifically developed to edit the PVM and the user stories at the same time and to ensure maximum consistency between these two requirement–representation artefacts (Shafiee et al., 2019). Another possibility would be to include more formal requirement–representation artefact elements in user stories; this would reduce the appeal of these artefacts, which is primarily based on their simplicity in writing and understanding.

The present study has several limitations that should be addressed in future research. The decision to conduct this study in a single company allowed the researchers to control for numerous influencing factors but limited the generalisability of the results. In addition, the single organisational context and the similarity of the chosen PCS projects did not allow the researchers to detect potential contingencies due to specific contexts, which could have affected the effectiveness of the PCS development. Therefore, in order to enhance the external validity of the results and to identify relevant contingencies, future qualitative and quantitative studies should be performed in different research settings or using larger statistical samples.

CRediT authorship contribution statement

Sara Shafiee: Conceptualization, Methodology, Validation, Formal analysis, Investigation, Resources, Writing - original draft, Writing - review & editing, Project administration. **Yves Wautelet:** Conceptualization, Methodology, Validation, Formal analysis, Writing - original draft, Writing - review & editing, Project administration. **Lars Hvam:** Conceptualization, Methodology, Validation, Investigation, Writing - original draft, Supervision. **Enrico Sandrin:** Conceptualization, Methodology, Validation, Formal analysis, Writing - original draft, Writing - review & editing. **Cipriano Forza:** Methodology, Validation, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Aldanondo, M., Rouge, S., Ve, M., 2000. Expert configurator for concurrent engineering: Caméléon on software and model. J. Intell. Manuf. 11, 127–134. http://dx.doi.org/10.1023/A:1008982531278.

Ambler, S., 2002. Agile Modeling: Effective Practices for Extreme Programming and the Unified Process, Guide to the Unified Process Featuring UML, Java and Design Patterns. John Wiley & Sons.

Ambler, S.W., 2005. A manager's introduction to the rational unified process (RUP). http://www.ambysoft.com.

Appleton, B., Konieczka, S., Berczuk, S., 2003. Agile change management: From first principles to best practices. CM J.

Ardissono, L., Felfernig, A., Friedrich, G., Goy, A., Jannach, D., Petrone, G., Schafer, R., Zanker, M., 2003. A framework for the development of personalized, distributed web-based configuration systems. AI Mag. 24, 93–110. http://dx.doi.org/10.1609/aimag.v24i3.1721.

Ariano, M., Dagnino, A., 1996. An intelligent order entry and dynamic bill of materials system for manufacturing customized furniture. Comput. Electr. Eng. 22, 45–60. http://dx.doi.org/10.1016/0045-7906(95)00027-5.

Barker, V.E., O'Connor, D.E., Bachant, J., Soloway, E., 1989. Expert systems for configuration at digital: XCON and beyond. Commun. ACM 32, 298–318. http://dx.doi.org/10.1145/62065.62067.

Basili, V.R., Weiss, D.M., 1984. A methodology for collecting valid software engineering data. IEEE Trans. Softw. Eng. SE-10, 728–738. http://dx.doi.org/10.1109/TSE.1984.5010301.

Benbasat, I., Goldstein, D.K., Mead, M., 1987. The case research strategy in studies of information systems. MIS Q. 11, 369. http://dx.doi.org/10.2307/248684.

Boehm, B., 2012. Get ready for agile methods, with care. Int. J. Eng. Sci. Technol. 4, 23–29. http://dx.doi.org/10.1109/2.976920.

Boehm, B., Turner, R., 2005. Management challenges to implementing agile processes in traditional development organizations. IEEE Softw. 22, 30–39. http://dx.doi.org/10.1109/MS.2005.129.

Campanelli, A.S., Camilo, R.D., Parreiras, F.S., 2018. The impact of tailoring criteria on agile practices adoption: A survey with novice agile practitioners in Brazil. J. Syst. Softw. 137, 366–379. http://dx.doi.org/10.1016/j.jss.2017.12.012.

Campanelli, A.S., Parreiras, F.S., 2015. Agile methods tailoring–A systematic literature review. J. Syst. Softw. 110, 85–100. http://dx.doi.org/10.1016/j.jss.2015.08.035.

Cardozo, E., Araújo Neto, B., Barza, A., França, C., da Silvia, F., 2010. Scrum and productivity in software projects: A systematic literature review. In: 14th International Conference on Evaluation and Assessment in Software Engineering. EASE, pp. 1–4.

Cervone, H.F., 2011. Understanding agile project management methods using Scrum. OCLC Syst. Serv.: Int. Digit. Libr. Perspect. 27, 18–22. http://dx.doi.org/10.1108/10650751111106528.

Cho, J., 2009. A hybrid software development method for large-scale projects: Rational unified process with Scrum. Issues Inf. Syst. 10, 340–348.

Collaris, R.-A., Dekker, E., 2010. Scrum and RUP: A comparison doesn't go on all fours. In: Agile Record. pp. 62–65.

Coram, M., Bohner, S., 2005. The impact of agile methods on software project management. In: 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems. ECBS'05, pp. 363–370. http://dx.doi.org/10.1109/ECBS.2005.68.

Drury-Grogan, M.L., Conboy, K., Acton, T., 2017. Examining decision characteristics and challenges for agile software development. J. Syst. Softw. 131, 248–265. http://dx.doi.org/10.1016/j.jss.2017.06.003.

Dwivedi, R., 2013. Configuration issues and efforts for configuring agile approaches: Situational based method engineering. Int. J. Comput. Appl. 61, 23–27. http://dx.doi.org/10.5120/10021-4941.

Dybå, T., Dingsøyr, T., 2008. Empirical studies of agile software development: A systematic review. Inf. Softw. Technol. 50, 833–859. http://dx.doi.org/10.1016/j.infsof.2008.01.006.

Edmondson, A.M.Y.C., Mcmanus, S.E., 2007. Methodological fit in management field research. Acad. Manag. Rev. 32, 1155–1179. http://dx.doi.org/10.5465/AMR.2007.26586086.

Felfernig, A., Friedrich, G.E., Jannach, D., 2000. UML as domain specific language for the construction of knowledge-based configuration systems. Int. J. Softw. Eng. Knowl. Eng. 10, 449–469. http://dx.doi.org/10.1016/S0218-1940(00)00024-9.

Felfernig, A., Hotz, L., Bagley, C., Tiihonen, J., 2014. Knowledge-Based Configuration from Research to Business Cases. Morgan Kaufmann, Newnes, NWS, Australia, http://dx.doi.org/10.1016/B978-0-12-415817-7.00029-3.

Forza, C., Salvador, F., 2002a. Managing for variety in the order acquisition and fulfilment process: The contribution of product configuration systems. Int. J. Prod. Econ. 76, 87–98. http://dx.doi.org/10.1016/S0925-5273(01)00157-8.

Forza, C., Salvador, F., 2002b. Product configuration and inter-firm coordination: An innovative solution from a small manufacturing enterprise. Comput. Ind. 49, 37–46. http://dx.doi.org/10.1016/S0166-3615(02)00057-X.

Forza, C., Salvador, F., 2006. Product Information Management for Mass Customization: Connecting Customer, Front-Office and Back-Office for Fast and Efficient Customization. Palgrave Macmillan, New York, NY.

Friedrich, G., Jannach, D., Stumptner, M., Zanker, M., 2014. Knowledge engineering for configuration systems. In: Felfernig, A., Hotz, L., Bagley, C., Tiihonen, J. (Eds.), Knowledge-Based Configuration: From Research to Business Cases. Morgan Kaufmann, pp. 139–155. http://dx.doi.org/10.1016/B978-0-12-415817-7.00011-6.

Hanakawa, N., Okura, K., 2004. A project management support tool using communication for agile software development. In: 11th Asia-Pacific Software Engineering Conference. pp. 316–323. http://dx.doi.org/10.1109/APSEC.2004.8.

Haug, A., Hvam, L., 2007. The modelling techniques of a documentation system that supports the development and maintenance of product configuration systems. Int. J. Mass Cust. 2, 1–18. http://dx.doi.org/10.1504/IJMASSC.2007.012810.

Haug, A., Shafiee, S., Hvam, L., 2019. The costs and benefits of product configuration projects in engineer-to-order companies. Comput. Ind. 105, 133–142. http://dx.doi.org/10.1016/j.compind.2018.11.005.

Heiskala, M., Paloheimo, K., Tiihonen, J., 2007. Mass customization with configurable products and configurators: A review of benefits and challenges. In: Mass Customization Information Systems in Business. IGI Global, Finland, pp. 75–106. http://dx.doi.org/10.4018/978-1-59904-039-4.ch001.

Heiskala, M., Tiihonen, J., Paloheimo, K., 2005. Mass customization of services: Benefits and challenges of configurable services. in: Frontiers of E-Business Research. FeBR 2005, Tampere, Finland, pp. 206–221.

Hollway, W., Jefferson, T., 2000. Doing Qualitative Research Differently: Free Association, Narrative and the Interview Method. Routledge, London, UK.

Hvam, L., Haug, A., Mortensen, N.H., Thuesen, C., 2013. Observed benefits from product configuration systems. Int. J. Ind. Eng.: Theory Appl. Pract. 20, 329–338.

Hvam, L., Mortensen, N.H., Riis, J., 2008. Product Customization. Springer-Verlag, Berlin Heidelberg, Germany, http://dx.doi.org/10.1007/978-3-540-71449-1.

Iacob, I., 2008. Extreme programming and rational unified process – Contrasts or synonyms? J. Inf. Syst. Oper. Manage. 2, 122–134.

Kamis, A., Koufaris, M., Stern, T., 2008. Using an attribute-based decision support system for user-customized products online: An experimental investigation. MIS Q. 32, 159–177. http://dx.doi.org/10.2307/25148832.

Kristjansdottir, K., Shafiee, S., Hvam, H., Forza, C., Mortensen, N.H., 2018. The main challenges for manufacturing companies in implementing and utilizing configurators. Comput. Ind. 100, 196–211. http://dx.doi.org/10.1016/j.compind.2018.05.001.

Kruchten, P., 2007. The Rational Unified Process: An Introduction, third ed. Addison-Wesley Professional.

Larman, C., 2004. Agile and Iterative Development: A Manager'S Guide, Communication. Addison-Wesley Professional.

Larman, C., Vodde, B., 2013. Scaling agile development: Large and multisite product development with large-scale Scrum. CrossTalk 9, 8–12.

Miller, K.W., Larson, D.K., 2005. Agile software development: Human values and culture. IEEE Technol. Soc. Mag. 24, 36–42. http://dx.doi.org/10.1109/MTAS.2005.1563500.

Moe, N.B., Dingsøyr, T., Dybå, T., 2010. A teamwork model for understanding an agile team: A case study of a Scrum project. Inf. Softw. Technol. 52, 480–491. http://dx.doi.org/10.1016/j.infsof.2009.11.004.

Noordeloos, R., Manteli, C., Vliet, H. Van, 2012. From RUP to Scrum in global software development: A case study. In: 2012 IEEE Seventh International Conference on Global Software Engineering. pp. 31–40. http://dx.doi.org/10.1109/ICGSE.2012.11.

Paetsch, F., Eberlein, A., Maurer, F., 2003. Requirements engineering and agile software development. In: 12th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises. pp. 308–313. http://dx.doi.org/10.1109/ENABL.2003.1231428.

Rising, L., Janoff, N.S., 2000. The Scrum software development process for small teams. IEEE Softw. 17, 26–32. http://dx.doi.org/10.1109/52.854065.

Rubin, K.S., 2012. Essential Scrum: A Practical Guide to the Most Popular Agile Process. Addison-Wesley.

Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. 14, 131–164. http://dx.doi.org/10.1007/s10664-008-9102-8.

Runeson, P., Host, M., Rainer, A., Regnell, B., 2012. Case Study Research in Software Engineering: Guidelines and Examples. John Wiley & Sons.

Sandrin, E., 2017. Synergic effects of sales-configurator capabilities on consumer-perceived benefits of mass-customized products. Int. J. Ind. Eng. Manage. 8, 177–188.

Sandrin, E., Trentin, A., Grosso, C., Forza, C., 2017. Enhancing the consumer-perceived benefits of a mass-customized product through its online sales configurator. Ind. Manage. Data Syst. 117, 1295–1315. http://dx.doi.org/10.1108/IMDS-05-2016-0185.

Selic, B., 2009. Agile documentation, anyone? IEEE Softw. 26, 11–12.

Shafiee, S., 2017. Conceptual Modelling for Product Configuration Systems. Technical University of Denmark, Denmark.

Shafiee, S., Friis, S.C., Lis, L., Harlou, U., Wautelet, Y., Hvam, L., 2019. A database administration tool to model the configuration projects. In: IEEE International Conference on Industrial Engineering and Engineering Management 2019-Decem. pp. 341–345. http://dx.doi.org/10.1109/IEEM.2018.8607654.

Shafiee, S., Hvam, L., Bonev, M., 2014. Scoping a product configuration project for engineer-to-order companies. Int. J. Ind. Eng. Manage. 5, 207–220.

Shafiee, S., Hvam, L., Haug, A., Dam, M., Kristjansdottir, K., 2017. The documentation of product configuration systems: A framework and an IT solution. Adv. Eng. Inform. 32, 163–175. http://dx.doi.org/10.1016/j.aei.2017.02.004.

Shafiee, S., Kristjansdottir, K., Hvam, L., 2016. Industrial experience from using the CPM procedure for developing, implementing and maintaining product configuration systems. In: 18th International Conference on Industrial Engineering. IJIE 2016, Seoul, South Korea.

Shafiee, S., Kristjansdottir, K., Hvam, L., Forza, C., 2018. How to scope configuration projects and manage the knowledge they require. J. Knowl. Manage. 22, 982–1014. http://dx.doi.org/10.1108/JKM-01-2017-0017.

Shuja, A.K., Krebs, J., 2007. IBM Rational Unified Process Reference and Certification Guide: Solution Designer (RUP). IBM Press.

Studer, R., Benjaminsc, V.R., Fensela, D., 1998. Knowledge engineering: Principles and methods. Data Knowl. Eng. 25, 161–197. http://dx.doi.org/10.1016/S0169-023X(97)00056-6.

Suzić, N., Forza, C., Trentin, A., Anišić, Z., 2018a. Implementation guidelines for mass customization: Current characteristics and suggestions for improvement. Prod. Plan. Control 29, 856–871. http://dx.doi.org/10.1080/09537287.2018.1485983.

Suzić, N., Sandrin, E., Suzić, S., Forza, C., Trentin, A., Anišić, Z., 2018b. Implementation guidelines for mass customization: A researcher-oriented view. Int. J. Ind. Eng. Manage. 9, 229–243.

Trentin, A., Perin, E., Forza, C., 2011. Overcoming the customization-responsiveness squeeze by using product configurators: Beyond anecdotal evidence. Comput. Ind. 62, 260–268. http://dx.doi.org/10.1016/j.compind.2010.09.002.

Trentin, A., Perin, E., Forza, C., 2012. Product configurator impact on product quality. Int. J. Prod. Econ. 135, 850–859. http://dx.doi.org/10.1016/j.ijpe.2011.10.023.

Trentin, A., Perin, E., Forza, C., 2013. Sales configurator capabilities to avoid the product variety paradox: Construct development and validation. Comput. Ind. 64, 436–447. http://dx.doi.org/10.1016/j.compind.2013.02.006.

Usman, M., Soomro, T.R., Brohi, M.N., 2014. Embedding project management into XP, SCRUM and RUP. Eur. Sci. J. 10, 293–307. http://dx.doi.org/10.19044/esj.2014.v10n15p%25p.

Vlaanderen, K., Jansen, S., Brinkkemper, S., Jaspers, E., 2011. The agile requirements refinery: Applying SCRUM principles to software product management. Inf. Softw. Technol. 53, 58–70. http://dx.doi.org/10.1016/j.infsof.2010.08.004.

Vlietland, J., Van Solingen, R., Van Vliet, H., 2016. Aligning codependent Scrum teams to enable fast business value delivery: A governance framework and set of intervention actions. J. Syst. Softw. 113, 418–429. http://dx.doi.org/10.1016/j.jss.2015.11.010.

Voss, C., Tsikriktsis, N., Frohlich, M., 2002. Case research in operations management. Int. J. Oper. Prod. Manage. 22, 195–219. http://dx.doi.org/10.1108/01443570210414329.

Wautelet, Y., Heng, S., Kiv, S., Kolp, M., 2017. User-story driven development of multi-agent systems: A process fragment for agile methods. Comput. Lang. Syst. Struct. 50, 159–176. http://dx.doi.org/10.1016/j.cl.2017.06.007.

Wautelet, Y., Heng, S., Kolp, M., Mirbel, I., 2014. Unifying and extending user story models. In: Jarke, M., Mylopoulos, J., Quix, C., Rolland, C., Manolopoulos, Y., Mouratidis, H., Horkoff, J. (Eds.), Advanced Information Systems Engineering. Springer International Publishing, Cham, pp. 211–225. http://dx.doi.org/10.1007/978-3-319-07881-6_15.

Yin, R.K., 2009. Case Study Research: Design and Methods (Applied Social Research Methods). Sage, Thousand Oaks, CA.