NordDesign 2020
August 11-14, 2020
Kgs. Lyngby, Denmark

Scrum Training for Product Configuration Systems Development

Sara Shafiee[1] & Yves Wautelet[2]

[1]*Technical University of Denmark*
sashaf@dtu.dk

[2]*Centre for Information Management, KU Leuven*
yves.wautelet@kuleuven.be

1 Introduction

Traditional software methodologies such as Waterfall, Spiral, and rational unified process (RUP) propose development models based on a sequential series of activities and steps which are well defined, comprehensive, up-front planned; documented in- detail and extensively designed (Gandomani, Zulzalil, Abdul Ghani, Abu, & Parizi, 2015). Unlike traditional methods, Agile methods embrace change in user requirements and emphasize the customer-centric approach in software development (Rubin, 2012). Agile methods provide different life cycles, roles, and activities compared to the traditional methods (Gandomani et al., 2015) and have been introduced to overcome the traditional methods challenges (Boehm, 2012). One of the important characteristics of the Agile approach in software development is giving priority to people, their roles, and interactions rather than processes and tools (Conboy, Coyle, Xiaofeng Wang, & Pikkarainen, 2011). Due to this feature, people and their roles, responsibilities, and behaviours are the main roots of the differences between Agile and disciplined methods (Javdani Gandomani & Ziaei Nafchi, 2016).

Product configuration systems (PCSs) support decision-making processes in the sales and engineering phases of a product with respect to product features and costs (Hvam, Mortensen, & Riis, 2008; Sandrin, Trentin, & Forza, 2018). PCSs enable companies to propose alternatives to facilitate their sales and production process (Felfernig, Hotz, Bagley, & Tiihonen, 2014; Forza & Salvador, 2006). Though product configuration systems have many advantages (Zheng, Xu, Yu, & Liu, 2017); such as shorter lead time (Hvam, Haug, Mortensen, & Thuesen, 2013; Trentin, Perin, & Forza, 2012; Zheng et al., 2017), fewer errors (Heiskala, Paloheimo, & Tiihonen, 2007) increased ability to meet customers' requirements regarding product functionality (Forza & Salvador, 2002), the use of fewer resources (Forza & Salvador, 2006), optimised product designs (Gronalt, Posset, & Benna, 2007; Trentin et al., 2012), less routine work, and improved on-time delivery (Ardissono et al., 2003; Liu, Shah, & Schroeder, 2006; Squire, Brown, Readman, & Bessant, 2009).

Reviewing previous studies shows that inadequate and dysfunctional training makes agile transformation ineffective (Conboy et al., 2011; Vijayasarathy & Turk, 2012). However, training Agile methods during a course will lead to notable confusion and slow down the development progress (Rico & Sayani, 2009). Literature reports training as a critical factor for successful process improvement and without useful training, the improvement is not satisfactory (Niazi, Wilson, & Zowghi, 2006). Moreover, significant correlation between successful implementation of Agile methods and receiving training has been proven (Livermore, 2008).

This paper, as an exploratory case study research, evaluates the satisfaction and relevance of Scrum training in one case company specialised in PCS development projects; it investigates the training materials and evaluates the drawbacks and strength of the realized training methods through interviews. The selected company is relevant because of its experiences with PCS projects through various development methods (RUP, and Scrum). This company experienced using RUP for developing PCS projects for five years and their transition to Scrum around three years ago was more a revolution than evolution. Hence, the novelty brought by Scrum and all its benefits and challenges introduce a completely new way of working to the whole team. Moreover, the company involves the researchers to optimize capacity management through improving the Scrum performance. A qualitative case study method is employed. First, the Scrum artefacts for PCS projects are determined in detail and different training steps are introduced. Secondly, we asked the same respondents about the benefits and challenges they face during PCS project while using Scrum.

2 Literature

2.1 Scrum

Scrum is an agile software development methodology. The *Agile Manifesto* outlines the values and principles that should be supported by the various agile processes applied in software development. Agile methods have steadily gained popularity since their introduction in the early 2000s. Agile principles emphasise customer satisfaction, change and collaboration between domain experts and developers (Paetsch, Eberlein, & Maurer, 2003). Rubin (2012) highlighted that with an agile approach, the team starts by creating a product backlog, which is a prioritised list of the features and other capabilities that need to be developed. Guided by the product backlog, team members address the most important or highest priority items first; priority is based on various factors, but delivered business value is most often the first priority. When the team runs out of resources (such as time), any work that was not completed will be of lower priority than the completed work. As noted above, Scrum is an agile approach for developing innovative products and services (Rubin, 2012). Figure 1 illustrates the main framework of Scrum as software project management process.

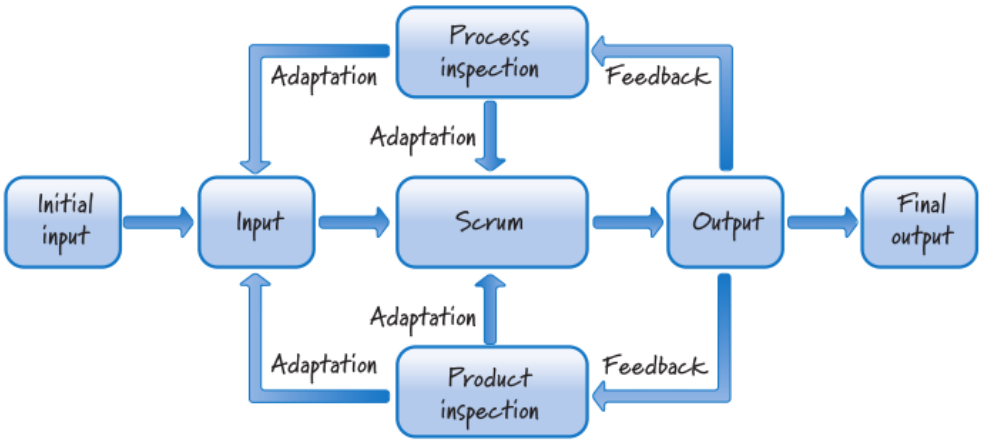

Figure 1. Scrum process model (Rubin, 2012)

In Scrum, a product owner acts as the single 'voice of the customer', collecting customer needs on a prioritised list of items, the product backlog (Cervone, 2011). The Scrum master arranges the daily Scrum meeting and location, tries to remove any production impediments, and serves as a coordinator between a Scrum team and other departments (Cho, 2009). The Scrum team is

responsible for developing software based on the sprint backlog (Rising & Janoff, 2000). Larman and Vodde (Larman & Vodde, 2013) introduced the product owner role as a means of coordinating multiple product owners.

The success of software development with Scrum is heavily dependent on the team. The team makes development decisions based on team consensus, and that team has more control over how the development is managed and completed (Cho, 2009). Overall, Scrum enables the developers to achieve better teamwork and a better communication, which results in higher-quality products (Hanakawa & Okura, 2004). Scrum facilitates cross-team coordination and collaboration (Vlaanderen, Jansen, Brinkkemper, & Jaspers, 2011). Vlietland et al. (2016) determined that Scrum improves coordination through additional events, such as interteam sprint planning meetings, interteam daily Scrums, interteam product refinements and interteam sprint reviews (Vlietland et al., 2016; Wautelet, Heng, Kiv, & Kolp, 2017).

In spite of the potential benefits of agile methods as Scrum, many organisations are reluctant to simply dispose of their conventional methods and replace them with agile methods. Agile in general and Scrum in particular have certain limitations, such as their light documentation and poor fit for large-scale projects (Collaris & Dekker, 2010; Usman, Soomro, & Brohi, 2014).

2.2 Engineering disciplines for PCS project development

The procedure proposed by Hvam et al. (2001) consists of seven disciplines, which aim to provide a structural approach to planning, developing, implementing and maintaining PCSs. The framework's disciplines include (1) development of the specification processes, (2) analysis of the product range, (3) object-oriented modelling, (4) object-oriented design, (5) programming of the PCS, (6) planning for implementation, and finally (7) planning for maintenance and further development (Hvam et al., 2008). For each of the phases, different artefacts are defined, and the utility is to represent/document relevant knowledge to fulfil the stage. The individual disciplines of the framework are shown in Figure 2. Product variant master (PVM) and class, responsibility, and collaboration (CRC) cards are examples of the main introduced artefacts.

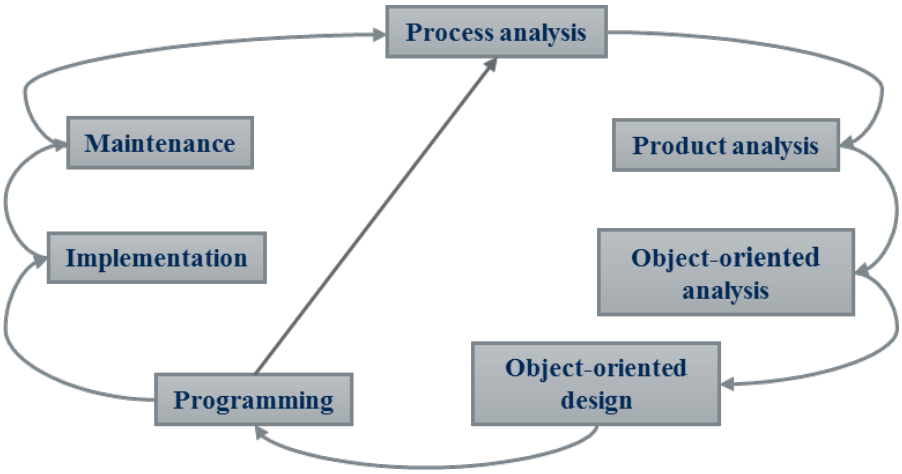

Figure 2. PCS project phases (Hvam et al., 2008)

The procedure emphasises the cross-disciplinary aspects of building and implementing PCSs, and it is derived from research and experience in different theoretical domains, which include the following:

- Mass customisation and modularisation of the products (Pine, 1993)
- Business process reengineering (Forza & Salvador, 2006)
- Product design and life cycle (Forza & Salvador, 2006)
- Architecture for building product models (Yang, Miao, Wu, & Zhou, 2009)
- Modelling techniques, such as object-oriented modelling (Felfernig, Friedrich, & Jannach, 2000)
- Software development, object-oriented analysis and design, knowledge representation and forms of reasoning for expert systems (Aldanondo, Rouge, & Ve, 2000; Felfernig et al., 2000)

As noted, within the specific disciplines for PCS development, custom artefacts for PCS knowledge modelling and visualisation are defined. The most important one in the context of this paper is the PVM, which enables a detailed analysis of the product range. For this purpose, the PVM is typically used together with CRC cards (Figure 3), both of which can be interfaced with classical UML artefacts. The PVM as presented by Hvam ((Hvam et al., 2008) displays product knowledge in a structured format, focusing on three different aspects: the customer's view, the engineering view and the production/part view. It is developed and used during the *product analysis*, *object-oriented analysis* and *object-oriented design* disciplines.

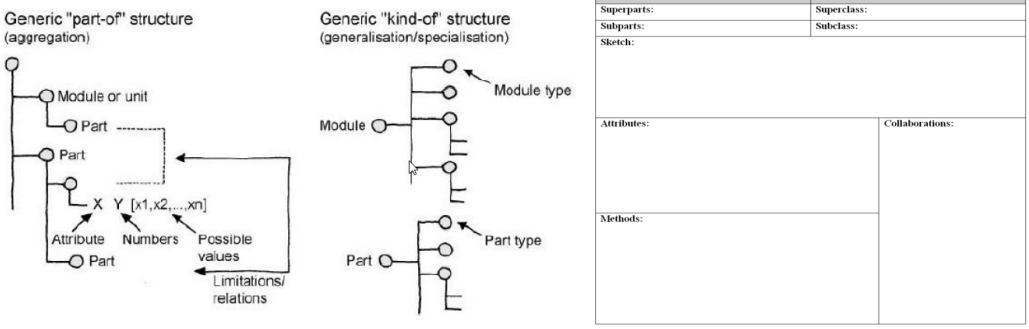

Figure 3. Structure of the PVM and CRC cards (Hvam et al., 2008)

2.3 Training and learning Scrum for IT development

There are few researches available on teaching Scrum and the training materials. Moreover, the most common introduced method for Scrum learning has been the traditional lecture model (Kruchten, 2011). However, it has been reported that the students should learn agile methods before trying to apply them during a capstone course (Rico & Sayani, 2009); hence, it will lead to notable confusion and slow down the development progress. Some of the researcher suggested games for Scrum training such as LEGO Bricks as a collaborative simulation of a Scrum development project. Paasivaara et al. (Paasivaara, Heikkilä, Lassenius, & Toivola, 2014) teach the Scrum s in practice by simulating development Sprints and actual development process, while incrementally planning and building a product of LEGO blocks. Some of the researchers investigate the experiences of introducing agile methods in four different academic programs (Melnik & Maurer, 2003) and the results demonstrates positive students' experiences while teaching agile methods in their software engineering courses.

3 Case study and findings

As an engineer-to-order company, the case company, which operates globally, specialises in catalyst production and process-plant technology. The study was limited to one case company in order to minimize the variable factors and keep the cultural and organisational influences static. The team members had from two to ten years of experience working with each of the methods. Two case projects that shared the following characteristics were selected:

1. PCS project dedicated to the products with almost the same level of complexity and from the same company,
2. use of Scrum approaches for PCSs,
3. potential access to management and senior experts at the companies,
4. development of all the selected PCSs in one specific software platform,
5. similar requirements,
6. similar users (engineers), the same IT team, and the involvement of similar tasks,
7. knowledge with similar setups, software and integrations.

Table 1 presents the case projects' detailed background information.

The complexity of the PCS projects is measured, by considering the cognitive complexity metrics which takes into the account the effort required to understand and modify the way in which the configuration problem has been modelled (Wang, 2006). Therefore, we assessed complexity in terms of two major PCS parameters: attributes and constraints (Brown, Keller, & Hellerstein, 2007; Shafiee, Hvam, Haug, Dam, & Kristjansdottir, 2017).

An open-ended interview technique (Yin, 2003) facilitated the collection of background data and consequently enhanced the richness of this study. The interviews were recorded, typewritten and analysed following the prescriptions suggested by (Eisenhardt, 1989; Yin, 2003). Among the 10 employees constituting the PCS development team, 5 have been selected for the interviews, based on criteria such as different years of experiences in Scrum, experience in PCS development, and different roles in the PCS team.

Analysing the background information and sprint, Figure 4 demonstrate the sprint procedure at the case company. This figure initiate with the business analyst and product owner and how the user stories are added to the backlog in the first step. Then in the next step, Scrum master add the story points to the user stories. Story point are defined based on hours and normally represents the amountof time need to be spent on each user story. This step is based on the experiences from previous similar user stories and also the experience of the main resource (developer) and it might change during the development phase. In the third step, sprint planning meetings are conducted. In sprint planning sessions, the whole scrum team will discuss about the backlog user stories and the doubts, clarifications and requirements. In step four, the tasks are broken down to smaller tasks (if required); then the team discuss technical design architecture, possible limitation and changes in the story points estimations. Moreover, developer gives his/her opinion on the amount of time he/she thinks it will take to complete for each task. In case of disagreement of 2 or more developers, that specific task is further discussed to take a certain amount of time and then everyone's opinion is asked again until a common estimate is reached.

Table 1. Background information for the PCS projects used as case studies.

Projects	***Case 1***	***Case 2***
Time frame (months)	11	8
Complexity of the configurator	Medium/high	Medium
No. of employees involved	5	6
Type of product to be modelled inside the PCS	A catalyst type	
Software platform (supporting the project)	A commercial configuration system	
Project participants (roles)	Project manager: 1 Development team: 2 Process engineer (business): 2	
Artefacts (main specifications)	Product goals and product backlog item (story) Product backlog and stakeholders' requirements (list of user stories) Testing (acceptance criteria in user stories)	
Planning approach	Daily Scrum Sprint planning Sprint review Feedback meetings	
Specific roles of meeting participants	Same as the project roles, plus: Product owner: 1 Scrum master: 1 Tester: 1	

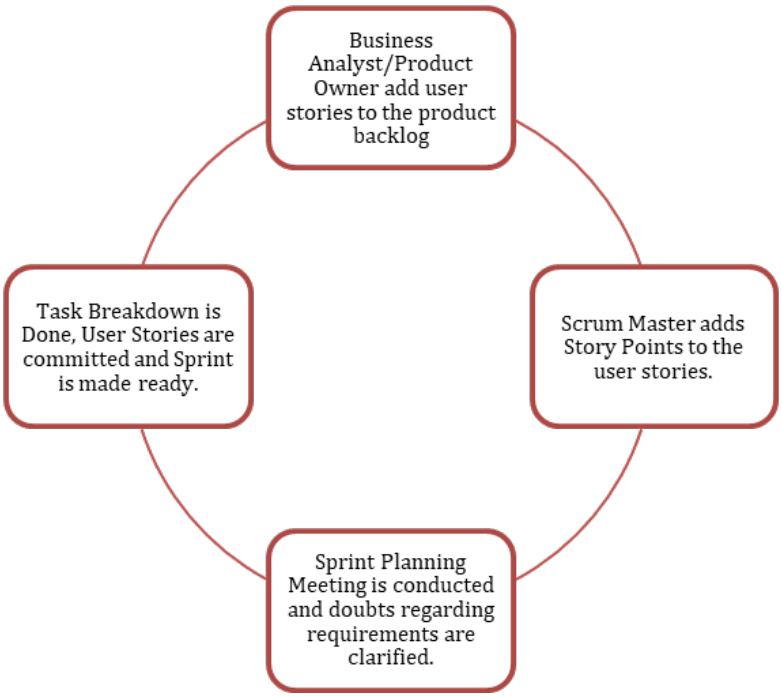

Figure 4. Sprint planning procedure at the case company

4 Discussions

4.1 Case study observation

In this section, we first present the observation results from the case study, which are summarised in Table 2. Documents, posters, and presentations are available for the training purposes in the PCS development team. Besides, 2 hours workshop is defined for every new member. We can divide the training at the case company into four main categories. As demonstrated in Table 2, first, all the relevant documents and the training materials are handed to the PCS team members. Secondly and simultaneously, the training is happening in the beginning as a short presentation. Third, the Scrum training from other experienced colleagues is a constant knowledge sharing during the projects implementation and sprints. These experiences are reported as lesson learned (what went bad, what went well) at the end of each sprint and are documented and handed over to new team members. Finally, the case company encourages the team to attend relevant courses and bring up new Scrum techniques for improving artefacts deigning.

Table 2. Standard training elements in the case companies

Training element	Training specification	Training time
Materials and documents	In the beginning of the employment	1 hours
Presentation- meetings- workshop	In the beginning of the employment	1.5 hours
Live training: During the project development	Knowledge sharing across the projects; lesson learned after each sprint.	1 hour discussion at the end of the Sprint (every 3 weeks)
Occasional presentations on new concepts: unit testing, test-driven development, etc.	Presentation based on the individual interests or courses: Once in the time	Presentation, discussions, and conclusion: 30 min per month

4.2 Interview results

The interviewees have some suggestions that might improve the training such as follow-up sessions and workshops to turn the purely theoretical training into a more practical training. It seems the training is not answering some of their questions and they mainly have been trained by practice and during 2-3 next months after receiving the training. Few of the employees also mentioned that the backlog planning and grooming need more clarity and effort of training as they observed the main challenges of the Scrum training for PCS related to planning phase. Mainly the activities included in planning of the backlog and sprints requires more efforts and it might be necessary to assist from tools and methods (if available) to have a more successful implementation.

Table 3. Results of open question interviews related to the training quality in PCS team at the case company.

Open Questions	
Based on the received training, is there any topic which were not relevant for you and your group? (you did not find useful, sufficient and relevant)	No comments
Based on the received training, is there any topic which you find relevant but not included in the training?	• Required visualization tools; • Roles description and responsibilities; • Time estimation for different user stories (development, testing and etc.); • Resources capacity measurements; • Tasks prioritization and task delegation; • Resolving blockers (impediments)

4.3 Summary of discussions

PCSs are considered as the most popular expert systems for mass customisation. Several challenges nevertheless appear while developing PCS. Therefore, researchers constantly investigate how development methods can facilitate PCS development. Scrum, as the most popular agile methodology, adequately addresses some of PCS projects challenges. Scrum can improve organizational efficiencies, resource constraints and customer satisfaction. Meanwhile, the training is a critical factor for successful Scrum process improvement. Organizations adopting Scrum might not be able to identify the core topics for successful Scrum training and the influence of their proper learning on Scrum performance; and poor literature is available on the subject.

The present study addresses these issues through a case study research using multiple data sources such as documentation, interviews, and participant-observations. First, we investigate the quality of training and the employees' assessments by running a first set of interviews within the selected case company. Secondly, we ask the employees about the quality of the current training at the case company for PCS projects. Moreover, we asked about the expectation for relevant topics to be included in the training. The study concludes with a reflection on future training challenges while developing PCS projects.

Observing the results from the the interviews, various benefits but also remaining challenges are reported by the employees. These challenges are grouped and listed as:

1. There is a lack of support for the visualisation and analysis of product knowledge (Product related challenge). As PCSs normally deal with complex product knowledge, specific tools for proper PCS engineering need to be used such PVM and CRC cards to visualize and confirm the knowledge.
2. Tacit knowledge and stakeholders' requirements cannot be fully represented with user stories, which are the basic artefacts for requirements representation (Knowledge acquisition challenges);
3. There is no structured framework for knowledge acquisition (Knowledge acquisition challenges);

4. Because of its complexity, poorly (agile) documentation of the product under development also negatively impacts the implementation of the PCS as well as the maintenance (Knowledge acquisition challenges);

5 Conclusion

The purpose of this paper is to investigate how to train PCS team members to adequately apply Scrum in PCS projects in order to build PCS of higher quality within a shorter time frame. This paper is an explanatory case study research to evaluate the training process in one case company. Based on the gathered data from case company and interview results, the Scrum training has been reported to be adequate while some drawbacks and challenges has been highlighted in the open questions. The team requires to be trained and provided with the required visualization tools. As PCS projects are complex and lots of dependencies exist in product structure, specific training is needed for time estimation for different user stories, roles description and responsibilities, resources capacity measurements, and tasks prioritization and task delegation. Moreover, impediments which mainly result from the dependency on business resources and requirements has been reported as a problem for which training could be provided.

The underlying study has been limited to one engineering company developing PCS projects. PCS can be categorized as complex and highly interactive IT systems. The in-depth study in only one case company gave the research team the opportunity to establish a deeper understanding of Scrum transition, its benefits and challenges, and the critical role of a high quality training for Scrum success in industry. It would be worthwhile for future research to further investigate this relation of Scrum training to the observed Scrum challenges in different case companies to provide more in-depth explanations and to offer specific solution. In particular, complex products development other than PCS could be studied to find out if the challenges linked to the complexity of the product lead to the same impact on Scrum application and thus training. Moreover, the relevant training suggestions for the observed Scrum challenges in PCS projects are relevant both for the practitioners and researchers.

References

Aldanondo, M., Rouge, S., & Ve, M. (2000). Expert configurator for concurrent engineering : Caméléon on software and model. Journal of Intelligent Manufacturing, 11(2), 127–134. doi:10.1023/A:1008982531278

Ardissono, L., Felfernig, A., Friedrich, G., Goy, A., Jannach, D., Petrone, G., … Zanker, M. (2003). A framework for the development of personalized, distributed web-based configuration systems. AI Magazine, 24(3), 93. doi:10.1609/aimag.v24i3.1721

Boehm, B. (2012). Get ready for agile methods, with care. International Journal of Engineering Science & Technology, 4(1), 23–29. doi:10.1109/2.976920

Brown, A. B., Keller, A., & Hellerstein, J. L. (2007). A Model of Configuration Complexity and its Application to a Change Management System Aaron. IEEE Transactions on Network and Service Management, 4(1), 13–27. doi:10.1109/TNSM.2007.030102

Cervone, H. F. (2011). Understanding agile project management methods using Scrum. OCLC Systems & Services: International Digital Library Perspectives, 27(1), 18–22. doi:10.1108/10650751111106528

Cho, J. (2009). A hybrid software development method for large-scale projects: Rational unified process with scrum. Issues in Information Systems, 10(2), 340–348.

Collaris, R.-A., & Dekker, E. (2010). Scrum and RUP: A comparison doesn't go on all fours. Agile Record, (1), 62–65. Retrieved from www.agilerecord.com

Conboy, K., Coyle, S., Xiaofeng Wang, L., & Pikkarainen, M. (2011). People over Process: Key Challenges in Agile Development. IEEE Software, 48–57. doi:10.1109/MS.2010.132

Eisenhardt, K. (1989). Building theories from case study. Academy of Management Review, 14(4), 532–550.

Felfernig, A., Friedrich, G. E., & Jannach, D. (2000). UML as domain specific language for the construction of knowledge-based configuration systems. International Journal of Software Engineering and Knowledge Engineering, 10(4), 449–469. doi:10.1016/S0218-1940(00)00024-9

Felfernig, A., Hotz, L., Bagley, C., & Tiihonen, J. (2014). Knowledge-based configuration from research to business cases. Newnes: Morgan Kaufman. doi:10.1016/B978-0-12-415817-7.00029-3

Forza, C., & Salvador, F. (2002). Managing for variety in the order acquisition and fulfilment process: The contribution of product configuration systems. International Journal of Production Economics, 76(1), 87–98. doi:10.1016/S0925-5273(01)00157-8

Forza, C., & Salvador, F. (2006). Product information management for mass customization: Connecting customer, front-office and back-office for fast and efficient customization. New York, NY: Palgrave Macmillan.

Gandomani, T. J., Zulzalil, H., Abdul Ghani, A. A., Abu, A. B., & Parizi, R. M. (2015). The impact of inadequate and dysfunctional training on agile transformation process: A grounded theory study. Information and Software Technology, 57(1), 295–309. doi:10.1016/j.infsof.2014.05.011

Gronalt, M., Posset, M., & Benna, T. (2007). Standardized Configuration in the Domain of Hinterland Container Terminals. In T. Blecker, G. Edwards, L. Friedrich, L. Hvam, & F. Salvador (Eds.), Series on Business Informatics and Application Systems Innovative Processes and Products for Mass Customization (Vol. 3, pp. 105–120). Berlin: GITO-Verlag.

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). doi:10.1109/APSEC.2004.8

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 (pp. 75–106). Finland: IGI Global. doi:10.4018/978-1-59904-039-4.ch001

Hvam, L., Haug, A., Mortensen, N. H., & Thuesen, C. (2013). Observed benefits from product configuration systems. International Journal of Industrial Engineering: Theory, Applications and Practice, 20(5–6), 329–338.

Hvam, L., Mortensen, N. H., & Riis, J. (2008). Product customization. Berlin Heidelberg, Germany: Springer-Verlag. doi:10.1007/978-3-540-71449-1

Javdani Gandomani, T., & Ziaei Nafchi, M. (2016). Agile transition and adoption human-related challenges and issues: A Grounded Theory approach. Computers in Human Behavior, 62, 257–266. doi:10.1016/j.chb.2016.04.009

Kruchten, P. (2011). Experience teaching software project management in both industrial and academic settings. 2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T), 199–208. doi:10.1109/CSEET.2011.5876087

Larman, C., & Vodde, B. (2013). Scaling agile development: Large and multisite product development with large-scale scrum. CrossTalk, 9, 8–12. Retrieved from

http://static1.1.sqspcdn.com/static/f/702523/22609354/1367558447003/201305-Larman.pdf
Liu, G. (Jason), Shah, R., & Schroeder, R. G. (2006). Linking Work Design to Mass Customization: A Sociotechnical Systems Perspective. Decision Sciences, 37(4), 519–545. doi:10.1111/j.1540-5414.2006.00137.x
Livermore, J. A. (2008). Factors that significantly impact the implementation of an agile software development methodology. Journal of Software, 3(4), 31–36. doi:10.4304/jsw.3.4.31-36
Melnik, G., & Maurer, F. (2003). Introducing Agile Methods in Learning Environments: Lessons Learned. In Extreme Programming and Agile Methods – XP/Agile Universe 2003 (pp. 172–184). doi:10.1007/978-3-540-45122-8_20
Niazi, M., Wilson, D., & Zowghi, D. (2006). Critical success factors for software process improvement implementation: An empirical study. Software Process Improvement and Practice, 11(2), 193–211. doi:10.1002/spip.261
Paasivaara, M., Heikkilä, V., Lassenius, C., & Toivola, T. (2014). Teaching students scrum using LEGO blocks. Companion Proceedings of the 36th International Conference on Software Engineering - ICSE Companion 2014, (June 2016), 382–391. doi:10.1145/2591062.2591169
Paetsch, F., Eberlein, A., & Maurer, F. (2003). Requirements engineering and agile software development. 12th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 308–313. doi:10.1109/ENABL.2003.1231428
Pine, B. J. (1993). Mass customization: The new frontier in business competition. Boston, Mass,: Harvard Business Review Press.
Rico, D. F., & Sayani, H. H. (2009). Use of Agile Methods in Software Engineering Education. In 2009 Agile Conference (pp. 174–179). doi:10.1109/AGILE.2009.13
Rising, L., & Janoff, N. S. (2000). The Scrum software development process for small teams. IEEE Software, 17(4), 26–32. doi:10.1109/52.854065
Rubin, K. S. (2012). Essential Scrum: A practical guide to the most popular agile process. Addison-Wesley.
Sandrin, E., Trentin, A., & Forza, C. (2018). Leveraging high-involvement practices to develop mass customization capability: A contingent configurational perspective. International Journal of Production Economics, 196, 335–345. doi:10.1016/j.ijpe.2017.12.005
Shafiee, S., Hvam, L., Haug, A., Dam, M., & Kristjansdottir, K. (2017). The documentation of product configuration systems: A framework and an IT solution. Advanced Engineering Informatics, 32, 163–175. doi:10.1016/j.aei.2017.02.004
Squire, B., Brown, S., Readman, J., & Bessant, J. (2009). The Impact of Mass Customisation on Manufacturing Trade-offs. Production and Operations Management, 15(1), 10–21. doi:10.1111/j.1937-5956.2006.tb00032.x
Trentin, A., Perin, E., & Forza, C. (2012). Product configurator impact on product quality. International Journal of Production Economics, 135(2), 850–859. doi:10.1016/j.ijpe.2011.10.023
Usman, M., Soomro, T. R., & Brohi, M. N. (2014). Embedding project management into XP, SCRUM and RUP. European Scientific Journal, 10(15), 293–307. doi:10.19044/esj.2014.v10n15p%25p
Vijayasarathy, L., & Turk, D. (2012). Drivers of agile software development use: Dialectic interplay between benefits and hindrances. Information and Software Technology, 54(2), 137–148. doi:10.1016/j.infsof.2011.08.003

Vlaanderen, K., Jansen, S., Brinkkemper, S., & Jaspers, E. (2011). The agile requirements refinery: Applying SCRUM principles to software product management. Information and Software Technology, 53(1), 58–70. doi: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. Journal of Systems and Software, 113, 418–429. doi:10.1016/j.jss.2015.11.010

Wang, Y. (2006). Cognitive complexity of software and its measurement. In Cognitive Informatics, 2006. ICCI 2006. 5th IEEE International Conference on (Vol. 1, pp. 226–235).

Wautelet, Y., Heng, S., Kiv, S., & Kolp, M. (2017). User-story driven development of multi-agent systems: A process fragment for agile methods. Computer Languages, Systems and Structures, 50, 159–176. doi:10.1016/j.cl.2017.06.007

Yang, D., Miao, R., Wu, H., & Zhou, Y. (2009). Product configuration knowledge modeling using ontology web language. Expert Systems with Applications, 36(3), 4399–4411. doi:10.1016/j.eswa.2008.05.026

Yin, R. K. (2003). Case Study Research: Design and Methods. CA; Newbury Park: Sage publications.

Zheng, P., Xu, X., Yu, S., & Liu, C. (2017). Personalized product configuration framework in an adaptable open architecture product platform. Journal of Manufacturing Systems, 43, 422–435. doi:10.1016/j.jmsy.2017.03.010