Software Quality Journal (2021) 29:337–379
https://doi.org/10.1007/s11219-021-09551-4

Scrum metaprocess: a process line approach for customizing Scrum

Halimeh Agh[1] · **Raman Ramsin**[1]

Accepted: 9 March 2021 / Published online: 7 April 2021

Abstract
Scrum is currently the most widely used agile methodology. However, it is regarded as a framework rather than a concrete process. Unfortunately, the resources available on Scrum do not explicitly define its variable parts and do not offer proper guidance on how to resolve those variabilities. Process (re)configuration is thus left to Scrum Retrospective sessions; this can delay the vital decisions that can significantly improve the process before problems arise. This paper aims to address the problems associated with configuring/reconfiguring Scrum by identifying all the variabilities (variation points) in the Scrum framework, along with the situations where a variation point can be resolved by one or more specific variants. We propose a Software Process Line (SPrL) approach for achieving this: we have represented the process variabilities of Scrum as a Scrum *metaprocess*, which acts as the core process of a generic SPrL for Scrum. The situations in which each variation point of the metaprocess can be resolved by a specific variant have been identified. The metaprocess has been implemented in the Medini-QVT tool, along with transformation rules that provide the means for automatic resolution of the variabilities. The validity of the metaprocess has been evaluated through an industrial case study, the results of which show that the metaprocess is applicable in real situations. Furthermore, the results indicate that the processes instantiated from the metaprocess can improve the existing processes by proposing specific practices for addressing their shortcomings.

Keywords Situational Method Engineering · Software Process Line · Scrum Framework · Situational Factor · Variability Resolution · Model Transformation

1 Introduction

Scrum (Schwaber & Sutherland, 2017) is not a concrete process: it is a process framework that encourages practitioners to iteratively evaluate and refine the process based on their own specific circumstances. Creating a specific Scrum process is a prerequisite for

✉ Raman Ramsin
ramsin@sharif.edu

Halimeh Agh
agh@ce.sharif.edu

[1] Department of Computer Engineering, Sharif University of Technology, Tehran, Iran

benefiting from its advantages. Although Scrum incorporates a potentially effective activity for frequent process review and revision, the "Sprint Retrospective", it is still preferable that an initial, suitably concrete version of the Scrum process be created before the start of every project. Failing to do so can result in unnecessary problems during the first sprints, raising the risk of development until the process stabilizes. These problems can be avoided if an initial version of the process is tailored based on the specific circumstances of the organization and/or the project at hand. The need for this initial method/process tailoring activity has been pointed out and/or addressed in several studies: situational engineering of agile methods based on situational factors has been investigated in Karlsson and Ågerfalk (2009) and Proba and Jung (2019); in Uikey and Suman (2016), a framework is proposed for tailoring agile methods before the start of product development; in Cram (2019), a survey of six software development companies that use Scrum or XP is reported, which shows that only one company uses agile methods in a "by the book" fashion, and the rest tailor their methods prior to enactment in order to increase their practicality; and in Masood et al. (2020), practitioners are encouraged to tailor Scrum roles, artefacts, and practices based on variations that the authors have identified by investigating how Scrum is actually used in practice.

Using a core process to show the variabilities of a set of software processes is a key aspect of Software Process Lines (SPrLs) (de Carvalho et al., 2014b). A SPrL is a specialized Software Product Line (SPL) in the context of process definition. In SPrL Engineering (SPrLE), the core process (also referred to as the *common architecture* and the *reference architecture*) is usually built by specifying the common and variable parts of existing processes; however, existing processes can have shortcomings and may therefore result in a less-than-appropriate core process. This problem can be avoided if existing mature processes, or process fragments extracted from reliable resources, are used for defining the core process.

The main contribution of this paper is an instantiable core process for Scrum. We have identified all the possible variabilities in the Scrum framework, as well as the situations where a variation point can be resolved with a specific variant. The approach consists of (1) a metaprocess to represent the variation points and their related variants; (2) for each variation point, the situations in which it is resolved by a specific variant; and (3) a set of transformation rules for automatic resolution of the variabilities based on the current project situation. The identified variabilities are of different types and granularities, namely, phase, activity, participant, input/output, strategy, practice, and technique. The proposed metaprocess and transformations required for resolving the variabilities have been implemented in the Medini QVT tool (Medini, QVT). In addition to industrial applications, the results of this research can also be used by researchers in academia (as discussed in Sect. 4). Researchers interested in software process engineering can use the metaprocess as a core asset for proposing novel approaches in fields such as Situational Method Engineering (SME) (Henderson-Sellers et al., 2014) and Software Process Improvement (SPI) (Pino et al., 2008).

The metaprocess has been validated through a case study. The results have shown that the proposed metaprocess is applicable in real situations. Furthermore, the results indicate that the processes instantiated from the metaprocess can improve the existing processes by proposing specific practices for addressing their shortcomings.

The rest of this paper is structured as follows: Sect. 2 introduces the metaprocess and its variabilities; Sect. 3 presents the best-fit situations for resolving the variabilities; Sect. 4 describes the case study conducted for validating the proposed metaprocess; Sect. 5

provides a brief survey of the related research; and Sect. 6 presents the concluding remarks and proposes directions for furthering this research.

2 Scrum metaprocess: variabilities

The variation points of the metaprocess should be identified along with the variants that can be used for resolving them. As mentioned in the Introduction section, variabilities are of seven types: phase, activity, participant, input/output, strategy, practice, and technique. In order to identify the different types of variabilities, we first need to identify the different types of process elements; for this purpose, we have investigated SPEM 2.0 (Software and System Process Engineering Metamodel) (OMG, 2008), since it is the de facto standard for modeling software development processes. In SPEM 2.0, there are three types of elements in the Method Content package: role, work product, and task. Furthermore, SPEM 2.0 incorporates the notions of phase, activity, and guidance for defining the lifecycle of a process. Therefore, variabilities in our work were initially classified as phase, activity, participant (role), input/output (work product), and guidance. However, we have identified three types of guidance in Scrum: strategy, practice, and technique; hence, "guidance" has been divided into three types, bringing the total number of types to seven. Technique and practice are often used interchangeably to specify a possible way (the how) for performing a unit of work; in order to differentiate between these two terms in this paper, practice has been used for referring to the agile practices recognized by the Agile Alliance, and everything else has been called technique. There are also variabilities that cannot be attributed to any of the above types; these variabilities have been grouped as "Other Variabilities". Each variation point is associated with zero or more variants and can be of three basic types:

- An optional variation point is a process element whose selection is dependent on a specific situation.
- An alternative-XOR variation point is the choice between one of two or more associated variants.
- An alternative-OR variation point is the selection (one or more) from two or more associated variants.

Figure 1 shows an overview of the proposed metaprocess. As shown in this figure, the Scrum metaprocess includes three classes of level-based variabilities: phase level, activity level, and role level; the remaining variabilities have been classified under "Other variabilities". At the phase level, only one variation point has been identified: maintenance with Kanban. At the activity level, all the potential variabilities of the eight activities defined in the Scrum framework (Rubin, 2012) have been identified; each activity includes various types of variabilities, including activity, participant, input/output, strategy, practice, and technique. At the role level, variabilities on who can fulfill the responsibilities of the three main roles defined in the Scrum framework (Rubin, 2012) have been specified; in addition, auxiliary roles can be defined for special situations. The variation points and their related variants are explained in the following sections. In order to identify the commonalities and variabilities of the Scrum metaprocess, various resources have been studied (e.g., Cohn, 2005, 2010; Rubin, 2012). In the process of identifying the variabilities, if a process element has been prescribed for all situations, then it has been categorized as a mandatory process element. As an example, Sprint Execution should be performed in all situations;

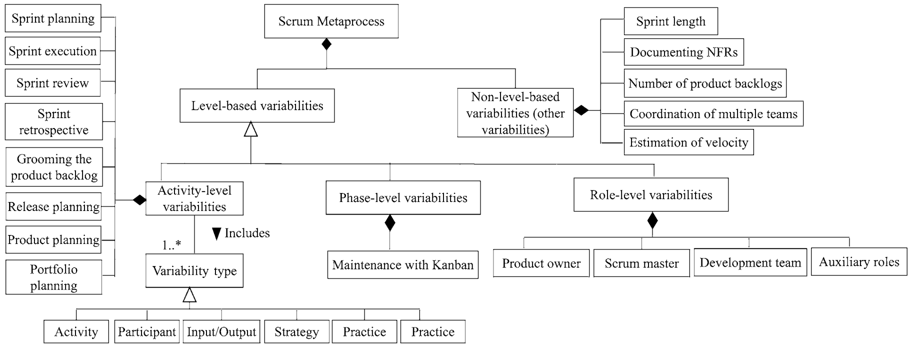

Fig. 1 Overview of the proposed Scrum metaprocess

therefore, it has been specified as a mandatory activity. However, there are process elements that are used in special situations; these process elements constitute the variable parts of the proposed metaprocess. For example, Portfolio Planning is only justifiable for organizations with multiple active, newly envisioned, or future products (Rubin, 2012); therefore, it is not essential for all organizations. Hence, Portfolio Planning has been classified as an optional activity in the proposed metaprocess.

2.1 Phase-level variabilities

The essential Scrum framework (Rubin, 2012) needs to be extended in order to be applicable to industrial projects; therefore, the framework has been extended as shown in Fig. 2. The graphical notation used in Fig. 2 is not a standard notation; however, it resembles the UML activity diagram. As explained in Fig. 2, the rectangle with rounded corners shows a mandatory phase/activity/task (akin to the action node in the UML activity diagram). The directional connection shows the sequence (unidirectional control flow) between tasks, and the connection without direction shows bidirectional control flows between tasks. The circular arrow shows that the activities grouped as the Development phase are conducted iteratively, so that after Sprint Retrospective, the next sprint starts with Sprint Planning. Sprint-level activities have been grouped into the *Development* phase. Scrum prescribes multilevel planning activities, including Portfolio Planning, Product Planning (Envisioning), Release Planning, Sprint Planning, and Daily Planning (Daily Scrum). The first three plannings are ongoing activities; we have therefore grouped them into the mandatory *Project Management* phase, with internal variation points that will be described further on.

Scrum does not include a phase specifically intended for maintenance purposes. Maintenance is a highly interrupt-driven activity, and Scrum has been recognized as unsuitable for this kind of work (Rubin, 2012). Several studies have shown that using Scrum for maintenance can pose severe challenges; as an example, the sheer volume of high-priority jobs emerging during maintenance can make the sprint backlog unreliable and prohibit the team(s) from meeting the sprint goal consistently (Heeager & Rose, 2015; Ibrahim et al., 2019). The results of a case study reported in Ahmad et al. (2016) point out that when Scrum is used for maintenance, the following negative outcomes can be expected: (1) lack of work visibility, (2) fluctuating task priorities, (3) over-commitment of sprints, (4) lack of communication and collaboration, and

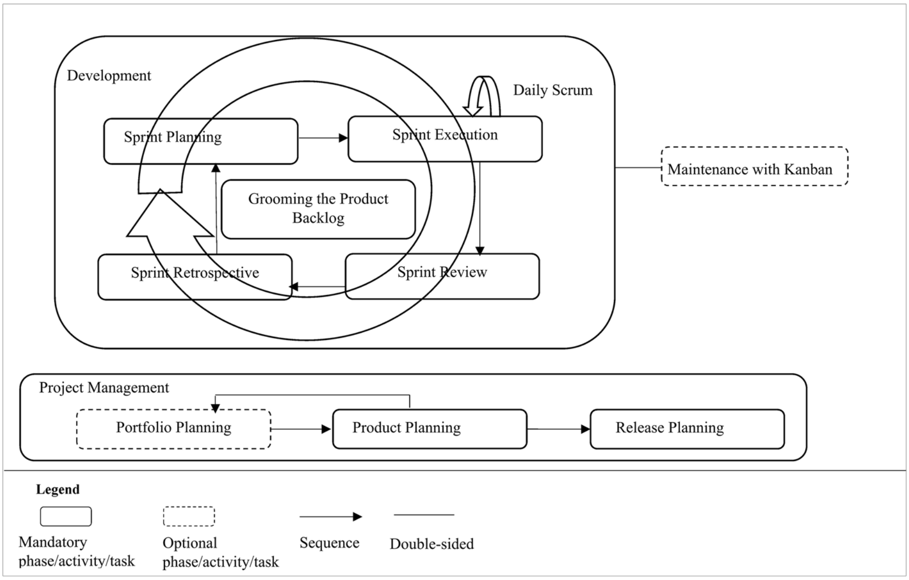

Fig. 2 Scrum Framework (adapted from (Rubin, 2012))

(5) lack of work synchronization. It has also been shown that these challenges can be suitably mitigated through Kanban (Ahmad et al., 2016; Pato et al., 2020; Seikola & Loisa, 2011; Sjøberg et al., 2012). Therefore, a *Maintenance with Kanban* phase has been added to the framework as an optional variation point. Using Kanban for maintenance purposes has been defined as an optional variation point; therefore, organizations can also use any other agile practice that they deem suitable for this purpose. It should be noted that development and maintenance can be (and usually are) performed in parallel, since a product being developed through an agile method is usually deployed into the operation environment in a gradual, release-by-release manner; as development proceeds beyond the first release, and new features are targeted for upcoming releases, parts of the system that have already been deployed are in active use, for which maintenance requests are constantly received and addressed. Differences between Scrum and Kanban have been discussed in Kniberg and Skarin (2010); we have used these discussions for determining the situations in which Kanban is known to fare better than Scrum, and vice versa, as shown in Table 1.

2.2 Activity-level variabilities

In this section, the different types of variabilities associated with each activity in the Scrum framework will be explained, starting with high-level planning activities. The notation that will be used throughout this paper for representing Scrum's commonalities and variabilities is shown in Fig. 3.

Table 1 Best-fit situations for using Scrum or Kanban (as alternatives)

	Scrum	Kanban
Best-fit situations	- Changes in feature priorities are allowed, but the permissible frequency is not as high as in event-driven environments.	- A high degree of change in feature priorities is allowed.
	- Goal alterations are not allowed throughout a sprint.	- Changes can be made at any time.
	- Timeboxing should be observed as a rule in delivering features.	- The product is delivered continuously.

2.2.1 Portfolio planning/management

Portfolio planning is an ongoing activity that can occur before or after product planning, and also at scheduled intervals for reviewing active products. Although this activity is optional, it is essential for organizations with multiple active, newly envisioned, or future products (Rubin, 2012). The process elements associated with this activity and their variabilities are explained in Table 2, and shown in Fig. 4.

2.2.2 Product planning (envisioning)

Product planning is a mandatory activity that begins whenever there exists an idea for a product that is consistent with the organization's strategic plan. The duration of this activity depends on the product that should be envisioned (Rubin, 2012). The process elements associated with this activity and their variabilities are explained in Table 3, and shown in Fig. 5.

2.2.3 Release planning

Release planning is an ongoing activity that is initially conducted after product planning; revising the release plan is performed during sprint review, but it can also coincide with sprint planning and/or sprint execution. There are two strategies for release planning: assigning specific features to specific iterations and not allocating work to specific

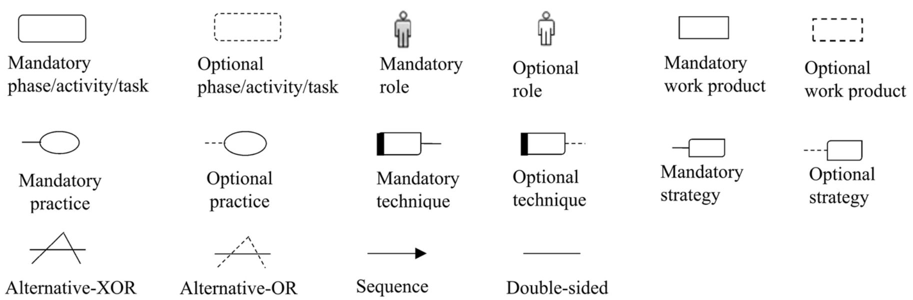

Fig. 3 Notation used for representing Scrum's commonalities and variabilities

Table 2 Process elements related to the portfolio planning activity

	Variants	Explanation
Inputs	New-product data In-process data	At least one of these inputs should be provided for starting the activity
Outputs	Portfolio backlog	----
	In-process products	----
Participants	Internal stakeholders	----
	Product owners of products	----
	Senior architects Technical leads	These roles participate in the planning when there are important technical constraints that should be considered in the portfolio-planning decisions
Activities	Scheduling	There are three strategies (Rubin, 2012): 1) Optimizing for lifecycle profits (across the entire portfolio): There are three alternative techniques: Shortest job first, High delay cost first, and Weighted shortest job first (Reinertsen, 2009; Rubin, 2012) 2) Calculating the cost of delay: There are two alternative techniques: Leffingwell model (Leffingwell, 2010), and General profile of the delay cost (Rubin, 2012) 3) Estimating for accuracy instead of precision: Relative estimation is preferred, typically by using T-shirt sizes as values
	Managing inflows	There are four strategies: Applying the economic filter, Balancing arrival rate with departure rate, Embracing emergent opportunities, and Planning for smaller and more frequent releases (Rubin, 2012)
	Managing outflows	There are three strategies: Focusing on idle work instead of idle workers, Establishing a WIP limit, and Waiting for a complete team (Rubin, 2012)
	Managing in-process products	Marginal economics can be used as the basis for managing active products

iterations. The first approach is time-consuming, but it can be beneficial when multiple teams work on a single project. Combining these two approaches typically yields a better solution (Cohn, 2005).

The duration of release planning depends on multiple factors such as product size, release risk, the participants' familiarity with the product, and the sprint length. While much time is spent on this activity early in the project, its duration will decrease as the project proceeds (Rubin, 2012). The process elements of this activity and their variabilities are explained in Table 4 and shown in Fig. 6.

2.2.4 Grooming the product backlog

Grooming the backlog is initially performed during product planning and release planning for creating and refining the high-level backlog initially produced; however, it is also performed continuously throughout the whole process. The process elements associated with this activity and their variabilities are explained in Table 5 and shown in Fig. 7.

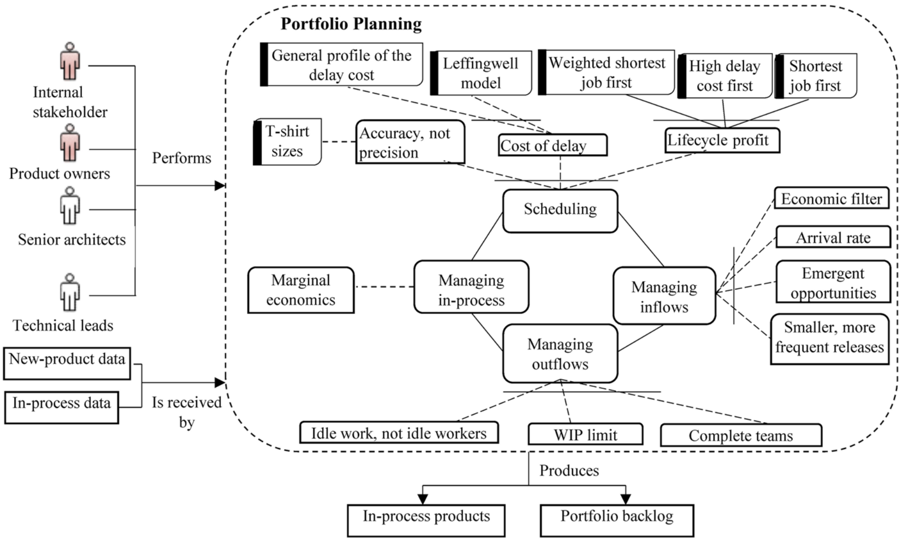

Fig. 4 Portfolio planning/management

2.2.5 Sprint planning

Sprint planning is an iterative activity performed at the beginning of each sprint. For each week in a sprint, sprint planning should take no longer than 2 hours (Rubin, 2012). The process elements of this activity and their associated variabilities are explained in Table 6 and shown in Fig. 8.

2.2.6 Sprint execution

Sprint execution is a mandatory activity conducted after sprint planning and before sprint review. The majority of the time spent on a sprint is allocated to this activity; it might take eight out of the 10 days during a 2-week-long sprint (Rubin, 2012). The process elements of this activity and their associated variabilities are explained in Table 7 and shown in Fig. 9.

2.2.7 Sprint review

This meeting is held after sprint execution and usually before sprint retrospective. The 1-hour-per-sprint-week rule (Rubin, 2012) is usually used for determining the length of the meeting. The process elements of this activity and their associated variabilities are explained in Table 8 and shown in Fig. 10.

Table 3 Process elements related to the product planning (envisioning) activity

	Variants	Explanation
Inputs	Initial idea	----
	Planning horizon	----
	Completion date	----
	Budget/resources	----
	Confidence threshold	----
Outputs	Product vision	----
	Product backlog	----
	Product roadmap	Produced if the optional product roadmap definition activity is performed
	Other artifacts	Outputs of optional activities (e.g., market research or competitive analysis)
Participants	Product owner	----
	Scrum master Development team	These roles are optional in initial envisioning; however, they should be included in any re-envisioning activity
	Internal stakeholder	One or more internal stakeholders usually collaborate with the product owner
	Other specialists	Involved if optional tasks are performed (e.g., market research)
Activities	Vision creation	The techniques useful for describing customer needs and product attributes are Personas and scenarios, use cases, and user stories (Pichler, 2009). There are also different formats for the vision document: elevator statement, product datasheet, product vision box, user conference slides, press release, and magazine review (Rubin, 2012)
	High-level product backlog creation	PBIs are often written as user stories (Cohn, 2010; Rubin, 2012). If the Scrum team is available during envisioning, the team and stakeholders write the stories; otherwise, the product owner and a few technical people interested in the product area write them. Practices commonly used for defining a PBI are INVEST (Independent, Negotiable, Valuable, Estimable, Small, and Testable), and Definition of Ready (Rubin, 2012)
	Product roadmap definition	Not necessary if there is only a single small release. The strategy of fixed periodic releases can be used, except in event-driven situations (Rubin, 2012)
	Other activities	Optional activities for achieving the target confidence threshold (Rubin, 2012); e.g., Market research, competitive analysis, and creating a rough business model

2.2.8 Sprint retrospective

This meeting usually occurs after the sprint review and before the next sprint. For 1-month sprints, this activity takes at most 3 hours (Schwaber & Sutherland, 2017). The process elements of this activity and their associated variabilities are explained in Table 9 and shown in Fig. 11.

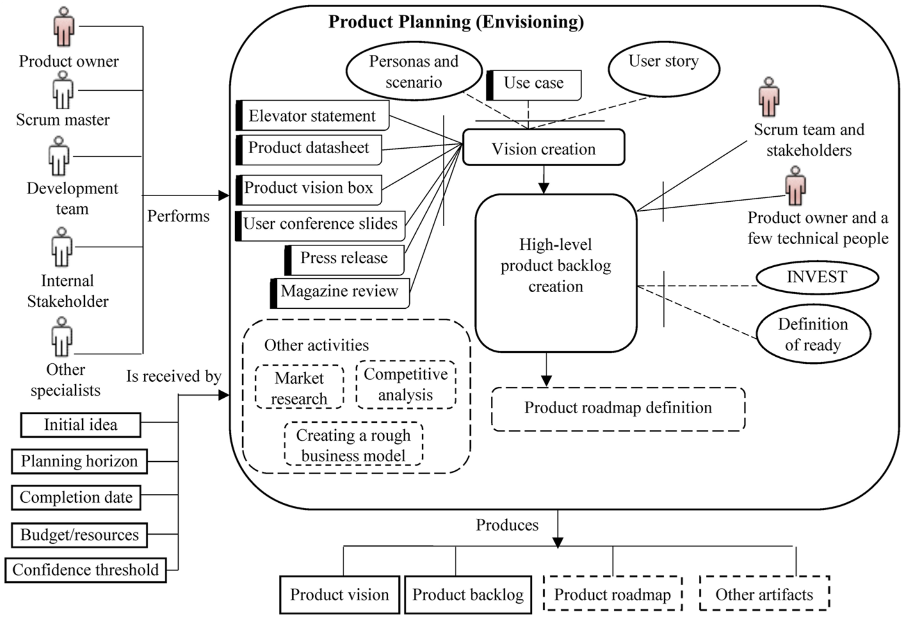

Fig. 5 Product planning (envisioning)

2.3 Role-level variabilities

While Scrum defines three main roles, there is a choice on who can fulfill their responsibilities. In addition, auxiliary roles can be defined for specific situations. These variabilities, shown in Fig. 12, are explained in the following sections.

2.3.1 Product owner

The variants for playing this role are as follows (Cohn, 2010; Diebold et al., 2015; Rubin, 2012):

- ○ **An analyst** who has adequate knowledge of the target product, along with the communication skills required.
- ○ **A development team member** with the necessary characteristics can act as both the product owner and a development team member.
- ○ **A project manager** with the skills and (domain) knowledge required.

2.3.2 Scrum master

The possible variants for playing this role are as follows (Cohn, 2010; Rubin, 2012; Yi, 2011):

Table 4 Process elements related to the release planning activity

	Variants	Explanation
Inputs	Product vision	----
	Product backlog	----
	Velocity	----
	Product roadmap	Input to the release planning activity, if produced during product planning
Outputs	Release plan	Includes MRFs (minimum releasable features), a rough estimation of the features deliverable by the release deadline (fixed date) or a rough delivery date for a given set of features (fixed scope), and optionally a sprint map; the sprint map is the output of Sprint Mapping (an optional activity)
Participants	Scrum team	----
	Internal stakeholders	----
Activities	Reviewing constraints	There are different combinations of Scope, Date, and Budget (the three important project constraints); however, the two most commonly used in Release Planning are: Fixed scope, and Fixed date. The fixed-scope strategy is appropriate for situations where the scope is more important than the time. The fixed-date strategy is appropriate in situations where features can be prioritized and a set of MRFs can be defined; this strategy is often more feasible (Cohn, 2010) If the fixed-scope strategy is selected, a Product burndown chart and/or a Product burnup chart can be used for communicating release progress. If the fixed-date strategy is selected, the product burnup chart can be used (Rubin, 2012). There are two types of product burndown chart: Burndown line chart, and Burndown bar chart (Cohn, 2005). A Parking-lot chart can also be used as a facilitation tool, and a Gantt chart (Cohn, 2005) is useful for showing the assignment of features to iterations
	Reviewing MRFs	—
	Grooming the product backlog	Since Grooming is an ongoing activity that extends beyond release planning, its process elements will be explained in a separate section
	Sprint mapping	This activity is not essential for single-team projects

○ **A technical lead** who helps the team implement its viewpoint, instead of dictating his/her decisions.
○ **Functional area managers or resource managers** with the skills required.
○ **A project manager** can act as a Scrum master provided that he/she does not direct the team or make decisions for it.
○ **A development team member** who has the necessary characteristics can act as both the Scrum master and a development team member; however, if a Scrum master has enough capacity, splitting her/his time among the teams can be a better solution.

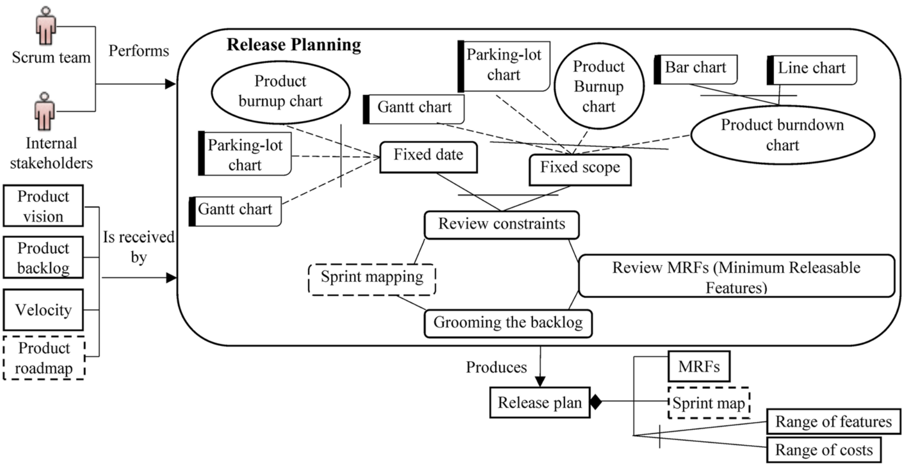

Fig. 6 Release planning

2.3.3 Development team

The size of a development team can be between 3 and 9 (Schwaber & Sutherland, 2017). There are various ways for organizing the structure of development teams (Rubin, 2012), including:

○ **Single Development Team:** if there is one small product to be built, forming just one cross-functional development team is typically enough.

○ **Feature Teams:** An organization should make every possible effort to structure the teams as feature teams. These teams should be cross-functional, encompassing all the skills necessary to develop the end-to-end functionality of each feature. Code ownership is shared among team members, and they collectively maintain code integrity.

○ **Component Teams:** Component teams are organized around layers or components of a product (Palomino et al., 2016). The deliverables of multiple component teams are integrated to make up a feature. Project teams are structured as component teams when parts of the code are reused by multiple feature teams, or when the organization/team is not yet able to adapt to the agile framework, or if sharing specialists across multiple teams is too difficult (Cohn, 2010; Larman, 2008).

○ **Coordinating multiple component teams with one feature team:** To coordinate multiple component teams, a feature team can be formed by selecting a representative from each component team to determine whether a feature is done.

2.3.4 Auxiliary roles

In addition to the main roles defined in Scrum, it is likely that a Scrum organization will define auxiliary roles as well, the two most common types of which are functional manager (resource

Table 5 Process elements related to the grooming activity

	Variants	Explanation
Inputs	Product backlog	----
Outputs	Groomed product backlog	----
Participants	Scrum team	----
Activities	Creating product backlog items (PBIs)	There are various practices and techniques for this purpose, including: 1) Exploratory testing (Cohn, 2010) 2) User story writing workshop (Rubin, 2012): There are two alternative strategies, bottom-up and top-down; the use of Personas is a beneficial practice 3) Story mapping (Rubin, 2012)
	Refining PBIs	Includes splitting, combining, and detailing the PBIs (Cohn, 2005)
	Estimating PBIs	Planning poker is a common practice for estimating PBI size (Cohn, 2010). Relative estimation is another practice, associated with three techniques: Silent grouping (Power, 2011), Affinity estimating (Sterling, 2008), and Bulk estimation (Greening, 2012). Silent grouping can complement planning poker in order to save time; affinity estimating and bulk estimation are recommended for larger numbers of PBIs. The two most common units for PBI size are Story Point and Ideal Day; story point is the unit usually preferred (Cohn, 2005)
	Prioritizing PBIs	There are four criteria for prioritizing PBIs: Value, Cost, Knowledge, and Risk (Cohn, 2005). Due to the complexity of estimating the financial return on a PBI, the desirability of PBIs for users is considered as an alternative criterion; there are two techniques for prioritization based on desirability: Kano analysis, and Relative weighting (Cohn, 2005)

manager) and project manager. The responsibilities of a project manager can be performed by any of the three main Scrum roles; however, a separate project manager is assigned when there is a large and complex development effort, or when Scrum is just applied to some parts of a large project (Canty, 2015; Rubin, 2012).

2.4 Other variabilities

There are variabilities in the Scrum framework that cannot be categorized as phase level, activity level, or role level; these variabilities are explained in the following sections.

2.4.1 Sprint length

The duration of sprints can vary from 1 week to 1 month (Schwaber & Sutherland, 2017); however, it is recommended that sprint durations be consistent, at least in the same release (Rubin, 2012; Schwaber & Sutherland, 2017). Sprint duration depends on several factors (Cohn, 2005), including release duration, risk level, ease of getting feedback, and iteration overhead.

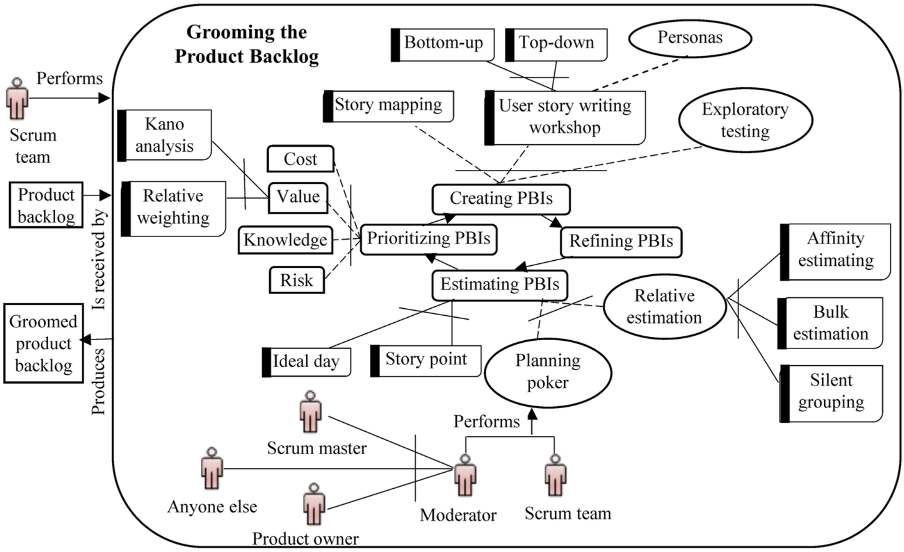

Fig. 7 Grooming the product backlog

2.4.2 Documenting Non-Functional Requirements (NFRs)

There are two ways for handling NFRs in Scrum: (1) writing NFRs as user stories and adding them to the product backlog and (2) integrating NFRs into the Definition of Done (DoD).

2.4.3 Number of product backlogs

One-product-one-product-backlog rule (Rubin, 2012) is usually used for determining the number of product backlogs. However, there are situations in which this rule is impractical (Rubin, 2012), including:

- If a large number of teams work independently on different areas of a large product, creating hierarchal backlogs is preferred.
- If there are multiple teams, each responsible for creating features consistent with its skill sets, creating team-specific views of a shared backlog is preferred.
- In some situations, there are multiple products, and therefore, multiple product backlogs, but limitations have forced the organization to have a single team working on all backlogs. In such situations, the product owner should assemble a prioritized set of PBIs from the multiple backlogs for every sprint.

Table 6 Process elements related to the sprint planning activity

	Variants	Explanation
Inputs	Product backlog	----
	Team velocity	----
	Team capabilities	----
	Constraints	----
	Initial sprint goal	----
Outputs	Sprint goal	----
	Sprint backlog	----
Participants	Scrum team	----
Activities	Determining capacity	There are two units for measuring capacity: Story-Points/Ideal-Days (the same units used for PBI size), and Effort-Hours (the same unit used for measuring sprint backlog tasks). After calculating the total capacity of a team, a buffer should also be reserved; there are three types of buffers: Feature buffer, Schedule buffer and Combined buffer (Cohn, 2005)
	Selecting PBIs	If a formal sprint goal is defined, PBIs are selected based on the goal; otherwise, they are selected from the top of the backlog up to the capacity of the team
	Acquiring confidence	There are two techniques for acquiring confidence: Using predicted velocity (Cohn, 2010), and creating the sprint backlog; the latter is preferred due to its reliability. It is recommended that the Definition of Done be used during this activity
	Refining sprint goal	----
	Finalizing commitment	----

2.4.4 Coordination of multiple teams

According to Nexus (Schwaber, 2015), if there are three to nine Scrum teams working on a single product backlog, the following process elements can be added to Scrum for coordinating the teams:

- ○ **Nexus sprint planning** is conducted before sprint planning to determine the PBIs for each team; three techniques can be used: establishing a common basis for estimates, adding detail to user stories sooner, and look-ahead planning (Cohn, 2005).
- ○ **Nexus sprint backlog** is the output of Nexus sprint planning and includes the PBIs selected by Scrum teams along with their dependencies.
- ○ **Nexus integration team** ensures the production of an integrated increment in every sprint; this team consists of the product owner, a Scrum master, and one or more Nexus integration team members.
- ○ **Nexus daily Scrum** is held before the daily Scrum activity; representatives from Scrum development teams attend this meeting to identify integration issues. The issues thus identified are fed to the daily Scrum in order to be addressed.
- ○ **Nexus sprint review** is performed by the teams in lieu of sprint reviews.
- ○ **Nexus sprint retrospective** includes the sprint retrospective activity performed by each Scrum team, along with two additional activities: identifying shared problems

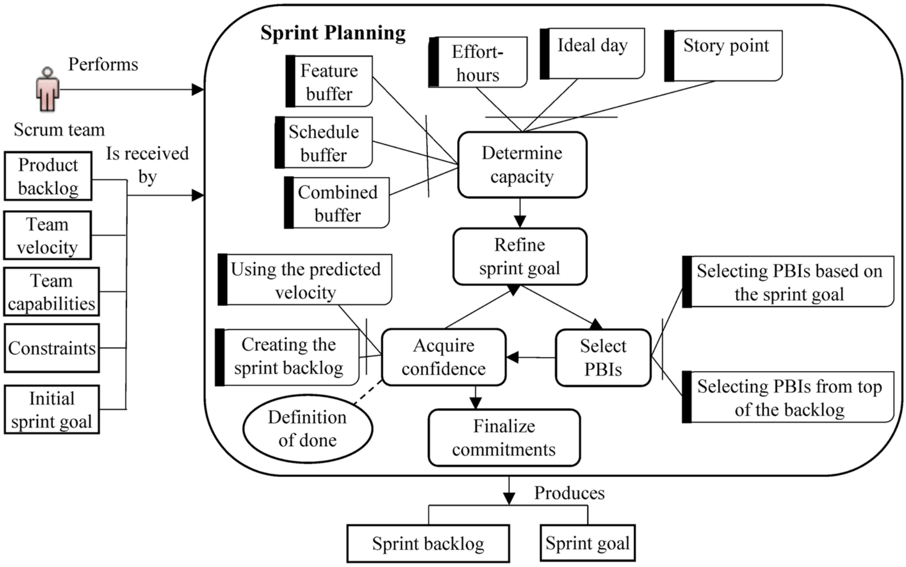

Fig. 8 Sprint planning

before the sprint retrospective, and discussing any actions needed for addressing the shared challenges after the sprint retrospective.

These process elements are added to the development phase of the Scrum framework, as shown in Fig. 13. There are other techniques for multi-team coordination that do not prescribe any specific range for the number of teams, including: Scrum of Scrums (Paasivaara et al., 2012) and Release Train (Rubin, 2012); *incorporating a feeding buffer into the plan* (Cohn, 2005) can be used as a sub-technique for the release train technique.

2.4.5 Estimation of velocity

There are three techniques for estimating velocity (Cohn, 2005), namely:

- **Using historical values** is preferred when the team has had previous experience in similar projects. There are two alternative techniques for estimating the velocity as a range: adding/subtracting points to/from the average velocity, and identifying the team's best and worst velocities over the past few months.
- **Running iterations** are used when there is no historical data. There are two alternative techniques: Using the cone of uncertainty, and Using the range of observed values; the second technique is only used when the team can run two or more sprints before estimating the velocity.
- **Forecasting** is used when it is not feasible to run sprints before estimating the velocity. It is done either by intuition, or by using the velocities of other teams.

Table 7 Process elements related to the sprint execution activity

	Variants	Explanation
Inputs	Sprint goal	----
	Sprint backlog	----
Outputs	Potentially shippable product increment	----
Participants	Scrum team	----
Activities	Task planning	----
	Flow management	----
	Daily Scrum	Although the maximum duration of this activity is 15 minutes, it may need to be adjusted for larger teams. The "Three Questions" is a practice usually used in this activity; the parking-lot chart can be used as a facilitation tool (McKenna, 2016). All Scrum team members attend this meeting; other people such as salespeople or developers from other projects can be invited, but they are only there to listen. The task board is frequently updated during this meeting
	Task performance	Five practices are commonly used for improving the Scrum teams' work during this activity (Cohn, 2010): Pair Programming (Arisholm et al., 2007; Chong & Hurlbutt, 2007; O'Donnell & Richardson, 2008; Padberg & Muller, 2003), Refactoring (Ge et al., 2006; Tingling & Saeed, 2007), Test-Driven Development (Causevic et al., 2011), Collective Ownership, and Continuous Integration (Warden & Shore, 2007)
	Communicating	A combination of the task board and the sprint burndown and/or burnup chart is commonly used for communicating progress; a Cumulative Flow Diagram (CFD) can be used for detailing a task/story. While the remaining effort is generally depicted in the sprint burndown chart in effort-hours, the work in the sprint burnup chart can be represented either in effort-hours or in story points

3 Situations for resolving Scrum variabilities

The variabilities of the Scrum framework were explained in Sect. 2. In this section, we will define the situations in which a variation point can be resolved by a specific variant. In SME, situational factors are usually used for expressing the specific characteristics of the project at hand and/or the organization's needs. We will use combination of situational factors for resolving the variabilities of the Scum metaprocess; however, we first need to specify the situational factors that are relevant to agile methodologies. The specific situations for resolving the variabilities are then defined in Sect. 3.2.

3.1 Situational factors for agile methodologies

In Clarke and O'Connor (2012), a reference framework is proposed for the situational factors affecting the software development process; we have used this framework to identify the situational factors that are relevant to agile methodologies and are useful for resolving

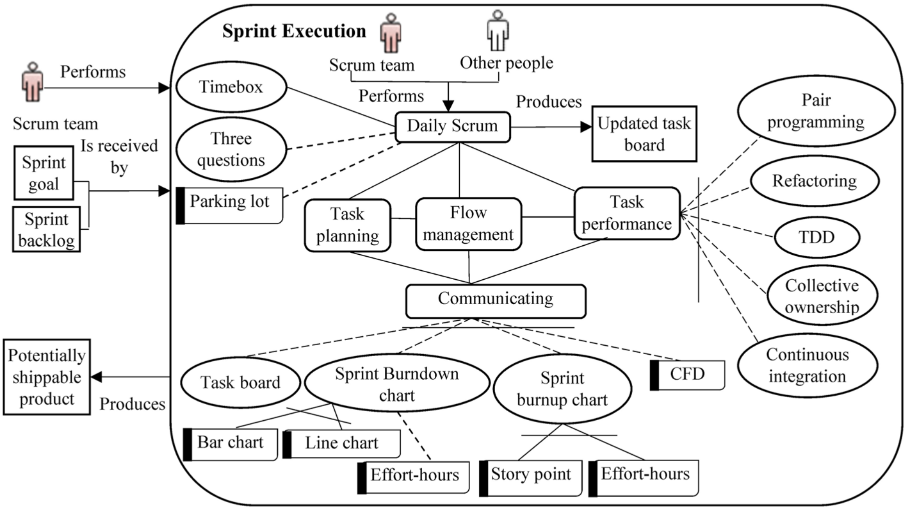

Fig. 9 Sprint execution

the variabilities identified in the proposed Scrum metaprocess. These situational factors were then refined and completed based on other resources (Abad et al., 2012; Ally et al., 2005; Campanelli & Parreiras, 2015; Clarke & O'Connor, 2012; Fitzgerald et al., 2006;

Table 8 Process elements of the sprint review activity

	Variants	Explanation
Inputs	Sprint goal	----
	Sprint backlog	----
	Potentially shippable product	----
Outputs	Groomed product backlog	----
	Updated release plan	----
Participants	Scrum team	----
	Internal stakeholders	----
	External stakeholders	Attendance of external stakeholders is not required at every Sprint Review, especially if the review involves internal discussions only
	Other internal teams	Other internal teams can attend the meeting to provide area-specific feedback or to synchronize their own work with the Scrum team
Activities	Summarizing	The sprint goal is presented, along with the PBIs associated with the goal, and a summary of the increment developed during the sprint
	Demonstrating	The increment developed during the current sprint is demonstrated
	Discussing	The current state of the product and the direction of the development are discussed
	Adapting	Based on the results of the discussion, participants may decide to create new PBIs, reprioritize the PBIs, or change/delete certain PBIs. This grooming might affect the release plan: it is possible that one or more of the release-plan variables (scope, date, and budget) be altered

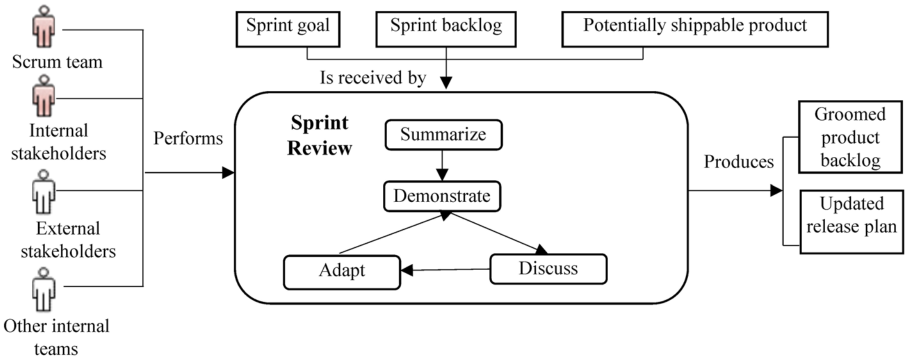

Fig. 10 Sprint review

Gill & Henderson-Sellers, 2006; Hoda et al., 2010; Hodgetts, 2004; Jyothi & Rao, 2012; Kruchten, 2013; Law & Charron, 2005; Lindvall et al., 2002; Rubin, 2012; Stankovic et al., 2013). The finalized list of situational factors is shown in Table 10, along with the additional resources that were used for refining or extending the factors; some factors have been redefined or combined in order to facilitate the resolution of variation points in the proposed metaprocess. The third column in Table 10 depicts the range of possible values for each situational factor; an explanation is also presented in the fourth column as a guideline for assigning specific values to each situational factor.

3.2 Specific situations for resolving variabilities

The specific situations for resolving each and every variability indicated in the Scrum metaprocess have been identified. However, for the sake of brevity, the best-fit situations for resolving the variabilities of "Grooming the product backlog" and sprint-level activities are presented herein, in Tables 11 and 12, respectively. The specific situations for resolving the remaining variabilities of the Scrum metaprocess are provided in Agh and Ramsin (2019).

3.3 Transformation rules

The time spent on defining/refining the target methodology can be reduced through automation; model-driven methods have shown great promise in this regard (Aleixo et al., 2010; Hurtado et al., 2013). Based on the situations identified for resolving the variabilities of the metaprocess, transformations have been implemented in the Medini-QVT tool to map each variation point to the appropriate variant(s). The metaprocess and situational factors are fed to these transformations as input models, and the target methodology is produced as output. Situational factors are modeled by instantiating the Software Process Context Metamodel (SPCM) (Hurtado et al., 2013) defined in the tool (Fig. 14).

Modeling the metaprocess is performed based on a metamodel defined for this purpose, which is an extended version of SPEM 2.0 (OMG, 2008). Although various extensions of SPEM 2.0 are already available, such as vSPEM (Martínez-Ruiz et al., 2011)

Table 9 Process elements of the sprint retrospective activity

	Variants	Explanation
Inputs	Focus	----
	Exercises	----
	Objective data	----
	Subjective data	----
	Insight backlog	Input to the activity if there are insights that have not been already addressed
Outputs	Improved actions	----
	Improved camaraderie	----
	Insight backlog	Produced if there are insights that will not be addressed in the upcoming sprint
Participants	Development team	----
	Facilitator	This role is usually performed by the Scrum master; however, any capable team member, or even an outside facilitator, can perform this role
	Product owner	If there are safety problems making the development team uncomfortable to speak freely in the product owner's presence, the product owner should not attend the meeting until the Scrum master resolves the issue
	Stakeholders/managers	The Scrum team decides whether these roles should attend the meeting
Activities	Setting the atmosphere	A set of ground rules should be established to help people safely express their views on improving the process. The atmosphere should be set so that everyone participates actively
	Creating a shared context	There are two common techniques for creating a shared context: Event Timeline, and Emotions Seismograph
	Identifying insights	Brainstorming and the Insight Backlog are two techniques commonly used for identifying the insights
	Determining actions	Dot Voting is commonly used for prioritizing the insights; then, the actions required for implementing high-priority insights are determined, to be performed during the next sprint
	Closing the retrospective	Certain tasks can be performed before ending the session, namely: reviewing the actions, thanking the participants, and collecting suggestions for improving the retrospective itself

and SmartySPEM (Junior et al., 2013), none of them support the multi-level variability modeling that is used in the proposed metaprocess for modeling variabilities at different levels, as shown in Sect. 2.2. The metamodel used for modeling the metaprocess is shown in Fig. 15. Figure 16 shows an example of the models produced by instantiating the metamodel. An example of the transformations implemented in Medini QVT is shown in Fig. 17. The transformations implemented in the tool are provided in Agh and Ramsin (2019), along with examples of applying them to the models. It should be noted that the variability models shown throughout Sect. 2 are the results of instantiating the proposed metamodel via Medini QVT. However, as shown in Fig. 16, the models created via this tool have a tree-like structure with nested nodes; understanding such models is difficult for the intended audience. Therefore, the models produced by the tool have been manually converted into more legible diagrams using a special notation, which has been shown in Fig. 3; the notation uses familiar elements from UML Activity Diagrams.

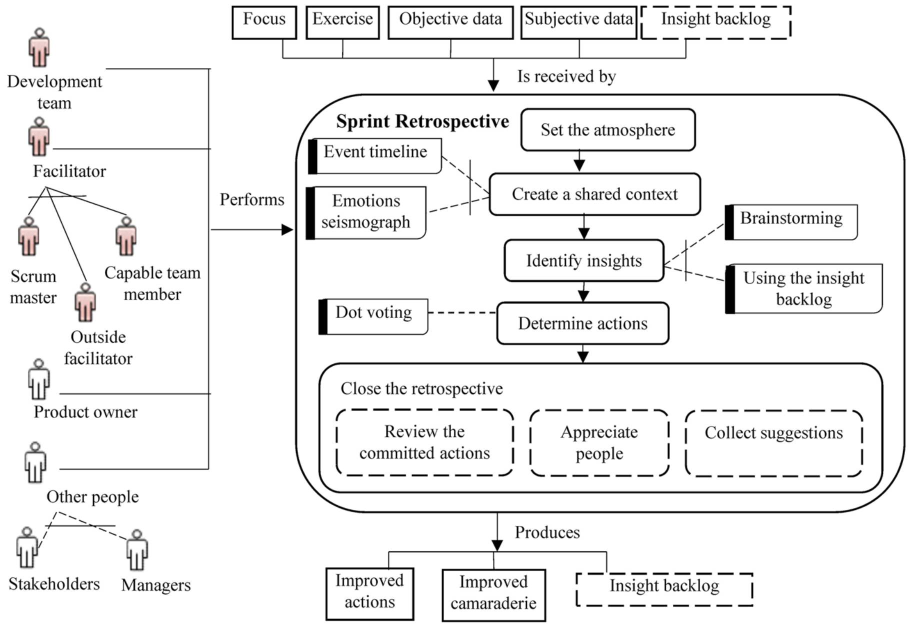

Fig. 11 Sprint retrospective

4 Case study

The validity of the proposed metaprocess has been assessed by conducting an industrial case study. The case study was conducted based on the recommendations of Runeson et al. (2012), as explained in the following sections.

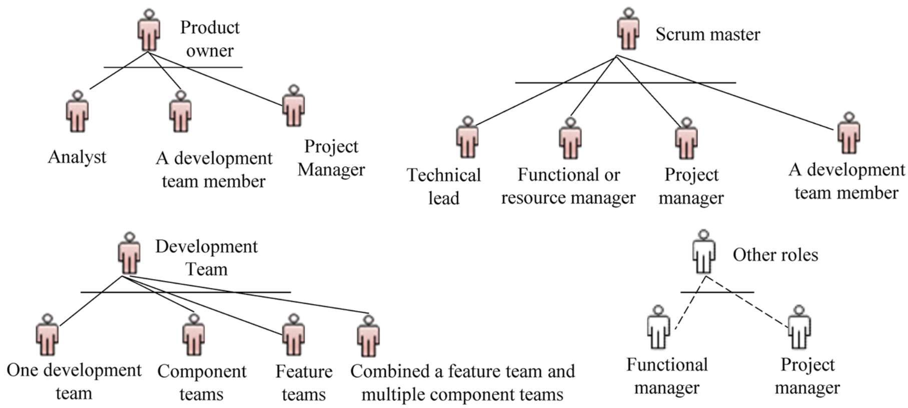

Fig. 12 Role-level variabilities

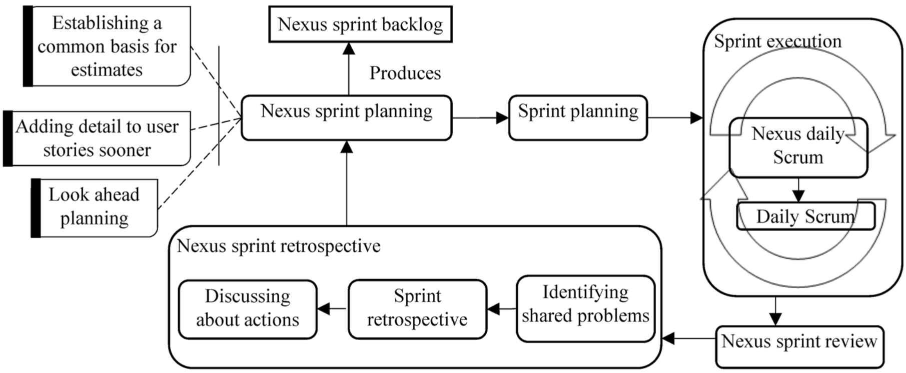

Fig. 13 Development phase of Scrum extended with Nexus process elements

4.1 Scope and objective

The study aims to investigate the effects of creating a specific process by using the proposed metaprocess in a software development organization. The study specifically aims to identify the applicability of the proposed approach in real situations as well as its effect on improving the processes already being used in organizations.

4.2 Research questions

The objective was refined into the following set of research questions:

- RQ1. Can the proposed Scrum metaprocess be used for building specific processes in real situations?
- RQ2. Can the specific processes produced from the proposed Scrum metaprocess improve the processes currently used in the organization?

To answer these research questions, we sought a case where Scrum was the organizational software process. It was also essential that people with enough knowledge about the processes being used in the organization had enough time to collaborate with us throughout the execution of the case study.

4.3 Case context

The case study was conducted based on the recommendations of Runeson et al. (2012). The study has been conducted in a medium-sized, Iranian software development company, which we will call A (the real name has been made anonymous). Company A has about 400 employees, is made up of different units, and uses Scrum as its organizational software process. The unit that was studied in this research develops banking software. To answer the research questions, three projects were selected in this unit, as shown in Table 13.

Table 10 Situational factors relevant to agile methodologies

Factor classification	Situational factors	Value	Explanation
Personnel	Number of teams	Normal/high	The number of development teams
	Culture (a two-valued factor) (Ally et al., 2005; Clarke & O'Connor, 2012; Lindvall et al., 2002)	Collaborative/non-collaborative–harmonious/disharmonious)	First value: the level of collaboration among team members to achieve the team's goal Second value: the level of interpersonal conflicts
	Experience (a two-valued factor) (Abad et al., 2012; Campanelli & Parreiras, 2015; Clarke & O'Connor, 2012; Fitzgerald et al., 2006)	Experienced/inexperienced–familiar/unfamiliar	First value: the level of developers' business knowledge. Second value: the level of developers' familiarity with the development methodology
	Cohesion (a two-valued factor) (Abad et al., 2012; Clarke & O'Connor, 2012; Kruchten, 2013; Stankovic et al., 2013)	Low/normal–normal/high	First value: the percentage of team members who have worked together in the past Second value: the rate at which people leave the team
	Skill & knowledge (Abad et al., 2012; Clarke & O'Connor, 2012; Hoda et al., 2010; Hodgetts, 2004; Stankovic et al., 2013)	Inadequate/adequate	The level of developers' technical knowledge and skill
	Commitment (Clarke & O'Connor, 2012; Stankovic et al., 2013)	Inadequate/adequate	The level of commitment to the project among team members
Requirements	Changeability (a two-valued factor) (Campanelli & Parreiras, 2015; Clarke & O'Connor, 2012; Lindvall et al., 2002)	Normal/high–normal/high	First value: the rate at which user and system requirements are changed Second value: the rate of scope creep
	Standards (Clarke & O'Connor, 2012)	Inadequate/adequate	The general quality of user requirements
Application	Degree of Risk (Fitzgerald et al., 2006; Rubin, 2012)	Normal/high	The level of project risks
	Complexity (Ally et al., 2005; Campanelli & Parreiras, 2015; Fitzgerald et al., 2006; Hodgetts, 2004; Rubin, 2012)	Normal/high	The level of application complexity
	Size (Abad et al., 2012; Clarke & O'Connor, 2012; Gill & Henderson-Sellers, 2006; Kruchten, 2013)	Normal/large	The relative application size
	Connectivity (Abad et al., 2012; Clarke & O'Connor, 2012)	Normal/high	The level of system dependence on existing/future systems
	Reuse (Clarke & O'Connor, 2012; Jyothi & Rao, 2012)	Normal/high	Extent of utilization of application components in other projects

Table 10 (continued)

Factor classification	Situational factors	Value	Explanation
	Deployment Profile (Clarke & O'Connor, 2012; Fitzgerald et al., 2006; Kruchten, 2013)	Normal/high	The number of different versions of the application to be deployed, or the number of applications to be deployed
	Quality (Abad et al., 2012; Ally et al., 2005; Clarke & O'Connor, 2012; Fitzgerald et al., 2006; Hodgetts, 2004)	Normal/high	The level of product quality required
Organization	Maturity (Clarke & O'Connor, 2012; Kruchten, 2013; Stankovic et al., 2013)	Inadequate/adequate	The level of organization maturity
	Management Commitment & Expertise (a two-valued factor) (Ally et al., 2005; Campanelli & Parreiras, 2015; Clarke & O'Connor, 2012; Stankovic et al., 2013)	(Inadequate/adequate–inadequate/adequate)	First value: The level of management's commitment to the project Second value: the level of management's knowledge and skill
	Facilities (Clarke & O'Connor, 2012; Stankovic et al., 2013)	Inadequate/adequate	The level of organization support for providing the facilities required, for example, physical environment
Operation	End-User Experience (Abad et al., 2012; Campanelli & Parreiras, 2015; Clarke & O'Connor, 2012)	Inadequate/adequate	The level of end-user familiarity with the application type
Business	Time to Market (Abad et al., 2012; Clarke & O'Connor, 2012; Hodgetts, 2004; Law & Charron, 2005)	Short/normal	The period of time available for building the first version of the shippable product
	External Dependencies (Clarke & O'Connor, 2012)	Normal/high	The number of the parties involved in building the product (multi-site development)
	Opportunities (Clarke & O'Connor, 2012)	Normal/high	The rate at which emergent opportunities occur
	Business Drivers (Abad et al., 2012; Clarke & O'Connor, 2012)	Financial considerations/marketing activities/minimizing costs/maximizing customer satisfaction	What is the crucial force behind the successful development of a project (the range of values for this factor can be extended)?
	Magnitude of Potential Loss (Abad et al., 2012; Clarke & O'Connor, 2012)	Normal/high	The effect of project failure on customer relations, financial health, competitive position, organizational reputation, organizational survival, or market share

Table 11 Situations to resolve variabilities of "Grooming the product backlog"

				Best-fit situations
Grooming the product backlog	Activity	Creating PBIs	Exploratory testing (Shah et al., 2014)	Personnel Experience = (Inexperienced, Familiar) or Application Complexity = High or Requirements Standards = Inadequate or End-user Experience = Inadequate
			User story writing workshop	Requirements Standards = Inadequate or Application Size = Large or Application Complexity = High; Personnel Culture = (Collaborative, Harmonious); Organization Facilities = Adequate
			Story mapping (Rubin, 2012)	Degree of Risk = High or Time to Market = Short or Personnel Experience = (Inexperienced, Familiar); Organization Facilities = Adequate
		Estimating PBIs	Relative estimation	Application Size = Large; Requirements Standards = Inadequate; Deployment Profile = High
			Planning poker (Mahnič & Hovelja, 2012)	Application Size = Normal; Personnel Culture = (Collaborative, Harmonious); Personnel Skill & Knowledge = Adequate
			Story point (Cohn, 2005)	Personnel Experience = (*, Familiar); Personnel Cohesion = (Low, High)
			Ideal day	Personnel Experience = (*, Unfamiliar); Personnel Cohesion = (Normal, Normal)
		Prioritizing PBIs (Cohn, 2005; Kaur & Kumar, 2015)	Cost	Business Drivers = Minimizing Costs
			Value	Business Drivers = Maximizing Customer Satisfaction
			Knowledge	Personnel Experience = (Inexperienced, Familiar) or Personnel Skill & Knowledge = Inadequate
			Risk (Reddaiah et al., 2013)	Degree of Risk = High or Magnitude of Potential Loss = High
	Strategies & practice	User story writing workshop	Personas	Personnel Experience = (Inexperienced, Familiar)
			Top-down	Personnel Experience = (Inexperienced, Familiar) or Application Complexity = High or Application Size = Large
			Bottom-up	Personnel Experience = (Experienced, Familiar); Application Complexity = Normal; Application Size = Normal
		Relative estimation	Affinity estimating (Sterling, 2008)	Application Size = Large
			Bulk estimation (Greening, 2012)	Application Size = Large; Time to Market = Short
			Silent grouping (Power, 2011)	Application Size = Large; Personnel Culture = (Collaborative, Harmonious)

Table 11 (continued)

			Best-fit situations
	Planning poker moderator	Scrum master	Number of Teams = High; Application Size = Large; Requirements Standards = Adequate
		Product owner	Number of Teams = Normal or Requirements Standards = Inadequate
		Anyone else	Application Size = Large; Number of Teams = High
	Value (Cohn, 2005)	Kano analysis	Personnel Skill & Knowledge = Inadequate

4.4 Study design

The study was exploratory and explanatory: it was exploratory in that it focused on confirming the benefits of the proposed metaprocess and finding new insights and ideas to improve it; it was explanatory in that it provided an explanation of a situation where the proposed metaprocess was applied as a new approach for creating specific processes.

The units of analysis were the software processes that were used in the different projects/units of the organization, the individual members at different positions in the organization, and the teams that these people were organized in. The studies focused on the experiences gained during the time that the teams were applying specific practices defined in the software process instantiated from the proposed metaprocess as compared to previous experiences in the same organization without the use of the metaprocess.

4.4.1 Subjects

In the case selected, there was no documented information about the software process used in the projects. Therefore, the subject sampling strategy was to hold interviews with a sample of people involved in the projects who had enough information about the process. In total, three people were interviewed who played “Project Manager” or “Product Owner” roles. To ensure the confidentiality of data and information, all the participants who attended the interviews were assured that the data would only be used for academic and research purposes.

4.4.2 Research methods

The main source of information in this investigation was semi-structured interviews. Interview instruments were constructed to focus on the areas of discussion. The instruments were also adapted as the interviews progressed to gain further information about the process used in the organization and the problems occurring during its execution. Three interview sessions were held with each subject. The length of each interview was approximately 1 hour. The interview instruments are provided in Agh and Ramsin (2019). In addition to the notes taken during the interviews, the interviews were also recorded in order to help prepare transcripts for later analysis.

Table 12 Situations to resolve the variabilities of sprint-level activities

				Best-fit situations
Sprint planning	Activity	Determine capacity	Relative weighting	Personnel Skill & Knowledge = Adequate
			Story point	Personnel Experience = (*,Familiar); Personnel Cohesion = (Low, High)
			Ideal day	Personnel Experience = (*,Unfamiliar); Personnel Cohesion = (Normal, Normal)
			Effort-hours	Personnel Cohesion = (Normal, Normal)
			Feature buffer (Cohn, 2005)	Time to Market = Short
			Schedule buffer (Cohn, 2005)	Degree of Risk = High or Application Complexity = High; Magnitude of Potential Loss = High
			Combined buffer (Cohn, 2005)	Time to Market = Short; Degree of Risk = High or Application Complexity = High; Magnitude of Potential Loss = High
		Select PBIs	Selecting PBIs based on the sprint goal	Personnel Cohesion = (Low, High); Degree of Risk = High
			Selecting PBIs from top of the backlog	Personnel Cohesion = (Normal, Normal); Requirements Standards = Adequate
		Acquire Confidence	Definition of Done	Magnitude of Potential Loss = High or Personnel Cohesion = (Low, High) or Degree of Risk = High or Application Quality = High
			Using the predicted velocity (Cohn, 2010)	Personnel Cohesion = (Normal, Normal); Application Complexity = Normal; Personnel Skill & Knowledge = Adequate
			Creating the sprint backlog	Personnel Cohesion = (Low, High); Application Complexity = High; Personnel Skill & Knowledge = Inadequate
Sprint Execution	Activity	Daily Scrum (Pauly et al., 2015)	Other people	Application Complexity = High or Degree of Risk = High or Personnel Experience = (Inexperienced, Familiar) or Personnel Skill & Knowledge = Inadequate
			Three questions	Personnel Experience = (*,Unfamiliar) or Time to Market = Short or Personnel Culture = (Non-collaborative,*)
			Parking lot chart (McKenna, 2016)	Application Complexity = High or Degree of Risk = High
		Task performance (Cohn, 2010)	Pair programming (Arisholm et al., 2007; Chong & Hurlbutt, 2007; O'Donnell & Richardson, 2008; Padberg & Muller, 2003)	Application Quality = High or Personnel Cohesion = (Low, High); Organization Facilities = Adequate; Personnel Culture = (Collaborative, Harmonious)

Table 12 (continued)

				Best-fit situations
			TDD (Causevic et al., 2011; Savoine et al., 2016)	Application Quality = High or Application Complexity = High or Degree of Risk = High or Requirements Standards = Inadequate; Personnel Skill & Knowledge = Adequate
			Refactoring (Ge et al., 2006; Hussain & Javed, 2015; Tingling & Saeed, 2007)	Application Reuse = High or Application Quality = High or Application Connectivity = High or Personnel Cohesion = (*,High)
			Collective ownership (Maruping et al., 2009)	Personnel Cohesion = (*,High) or Application Quality = High or Time to Market = Short; Personnel Skill & Knowledge = Adequate; Personnel Culture = (Collaborative, Harmonious)
			Continuous integration (Warden & Shore, 2007)	Deployment Profile = High or Time to Market = Short or Degree of Risk = High; Magnitude of Potential Loss = High or Application Connectivity = High
		Communicating	CFD (Power, 2014)	Requirements Changeability = (*,High); Personnel Experience = (*,Unfamiliar)
			Task board (Hajratwala, 2012)	Personnel Skill & Knowledge = Inadequate or Requirements Changeability = (High,*)
			Burndown chart	Requirements Changeability = (*,Normal)
			Burnup chart	Requirements Changeability = (*,High)
	Practice	Burndown chart	Effort-hours	Personnel Cohesion = (Normal, Normal)
		Burnup chart	Story point	Personnel Experience = (*,Familiar); Personnel Cohesion = (Low, High)
			Effort-hours	Personnel Experience = (*,Unfamiliar); Personnel Cohesion = (Normal, Normal)
Sprint review	Role		External stakeholders	Business Opportunities = High or Degree of Risk = High or Application Quality = High
			Other internal teams	External Dependencies = High or Application Complexity = High
Sprint Retrospective (Erdoğan et al., 2018; Jovanovic et al., 2015)	Role	Other people	Stakeholders	Personnel Experience = (Inexperienced, Familiar) or Business Opportunities = High
			Managers	Management Commitment & Expertise = (Adequate, Adequate)

Table 12 (continued)

			Best-fit situations
	Product owner		Personnel Culture = (Collaborative, Harmonious); Requirements Changeability = (High,*)
	Facilitator	Scrum master	Number of Teams = Normal or External Dependencies = Normal
		Outside facilitator	Application Complexity = High or Degree of Risk = High or Personnel Experience = (*,Unfamiliar)
		A capable team member	Application Size = Large; Number of Teams = High
Product	Insight backlog (Input)		Personnel Experience = (Inexperienced,*); Application Size = Large or Application Complexity = High
	Insight backlog (Output)		Personnel Experience = (Inexperienced,*); Application Size = Large or Application Complexity = High
Activity	Create a shared context	Event timeline	Application Complexity = High or Degree of Risk = High or Application Size = Large; External Dependencies = High
		Emotions seismograph	Management Commitment & Expertise = (Adequate, Adequate); Personnel Culture = (Collaborative, Harmonious)
	Identify insights	Brainstorming	Personnel Culture = (Collaborative, Harmonious); Management Commitment & Expertise = (Adequate, Adequate)
		Using the insight backlog	Personnel Experience = (Inexperienced,*); Application Size = Large or Application Complexity = High
	Determine actions	Dot voting	Personnel Culture = (Collaborative, Harmonious)
	Collect suggestions		Personnel Culture = (Collaborative, Harmonious); Management Commitment & Expertise = (Adequate, Adequate)
	Appreciate people		Management Commitment & Expertise = (Adequate, Adequate)
	Review the committed actions		Management Commitment & Expertise = (Adequate, Adequate)

4.4.3 Analysis

The audio recordings of the interviews were fully transcribed to identify problems in the processes being used in the projects. In order to apply the process instantiated from the metaprocess, a SPI (Software Process Improvement) roadmap was prepared to gradually implement the proposed practices. Based on the SPI roadmap created, a subset of the proposed practices, which was applicable during a 2-week sprint, was applied. At the end of the sprint, team members provided their feedback using a questionnaire; the questionnaire is provided in Agh and Ramsin (2019).

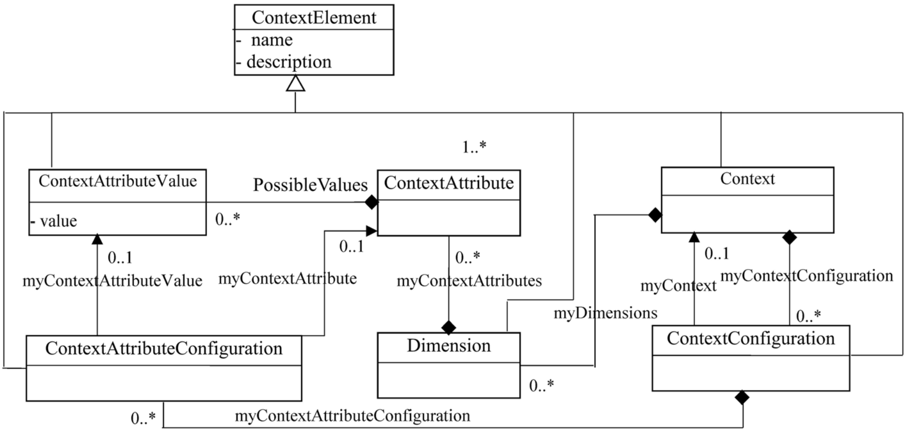

Fig. 14 Software Process Context Metamodel (SPCM) (Hurtado et al., 2013)

4.5 Study validity

The tactics used for reducing threats to validity are as follows:

- Theory triangulation: The viewpoints of the different roles involved were considered throughout the case study (project manager and product owner) as well as in filling out the questionnaire designed for collecting feedback about the proposed practices (project manager, product owner, and development team).
- Developing and maintaining a detailed case study protocol: A case study protocol was defined at the beginning of the study and was updated continuously throughout the study.
- Review of the designs, protocols, etc. by a peer researcher: The procedures selected for data collection and analysis were reviewed by a highly effective person (the second author).

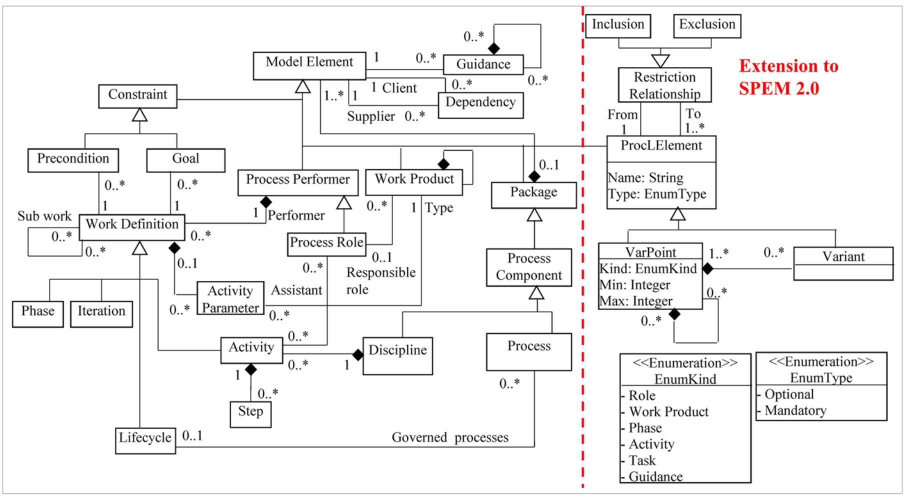

Fig. 15 Metamodel used for modeling the metaprocess

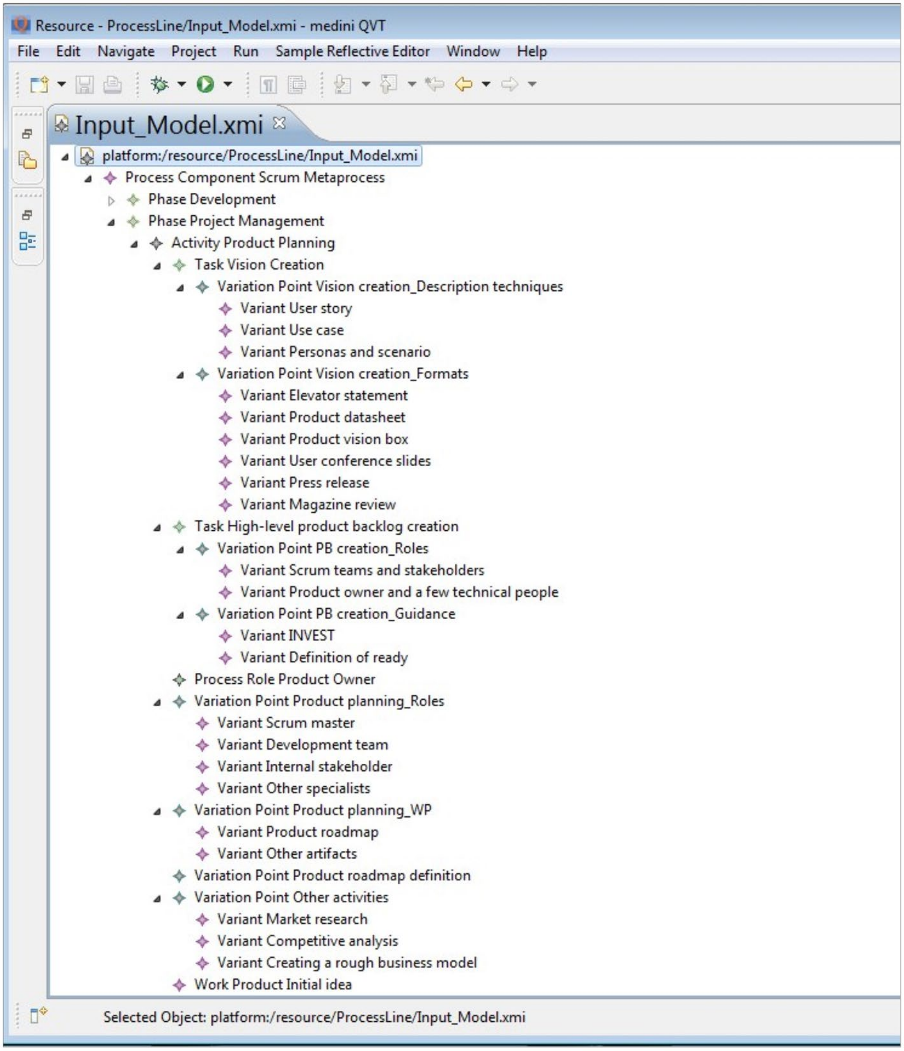

Fig. 16 An excerpt of the Scrum metaprocess defined in the tool

– Review of the collected data and obtained results by case subjects: The results of each interview session were reported back to the subjects via email and discussed in face-to-face conversations, and misinterpretations were resolved.

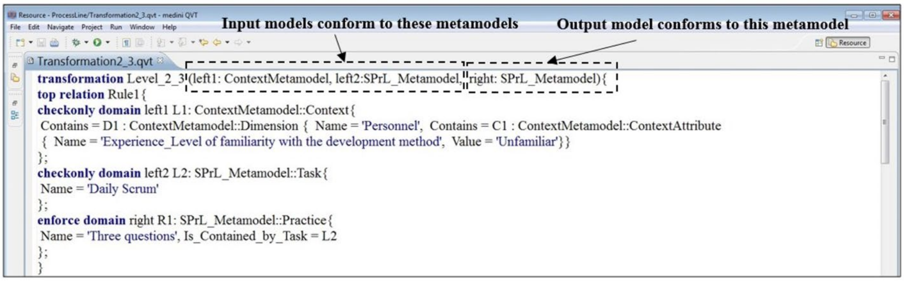

Fig. 17 Example of transformations implemented in Medini-QVT

4.6 Applying the metaprocess to the case study

In this section, the results of applying the metaprocess to the case study for constructing project-specific processes are presented. The answers to the research questions RQ1 and RQ2 are provided below.

4.6.1 RQ1-Building specific processes for real situations via the metaprocess

To answer this question, situational factors were given specific values, as shown in Table 14. The values were determined by considering the specifications of the projects and the organization; moreover, the subjects' knowledge and experience were used for this purpose through the second interview session. These values contributed to resolving the variabilities defined in the metaprocess. The resulting instantiated processes are too large to be reported herein; therefore, only the resolved "Grooming the product backlog" activity for project A.2 will be shown. The corresponding activity currently being used in project A.2 is shown in Fig. 18, and the instantiated process produced by resolving the variabilities is shown in Fig. 19. The complete set of existing and produced processes for the three projects is provided in (Agh & Ramsin, 2019). The subjects agreed that the produced processes were suitable for their projects and could potentially solve their problems; a list of the problem–solution pairs is provided in (Agh & Ramsin, 2019).

It should be noted that satisfying RQ1 is not a trivial task. An immature metaprocess can easily fail to satisfy this requirement. Two hypothetical cases in which a metaprocess would not be considered as applicable to real situations are provided below:

Table 13 Projects selected in case A

	Project selected	Number of development teams involved
A.1	Internet Banking Software	3
A.2	Improving the Capabilities of Smart Cards	1
A.3	Internet Store	1

Table 14 Values of situational factors in case A

Factor classification	Situational factors	A.1	A.2	A.3
Personnel	Number of teams	Normal	Normal	Normal
	Culture	(Collaborative, Harmonious)	(Collaborative, Harmonious)	(Collaborative, Disharmonious)
	Experience	(Experienced, Unfamiliar)	(Inexperienced, Familiar)	(Experienced, Familiar)
	Cohesion	(Normal, Normal)	(Normal, Normal)	(Normal, Normal)
	Skill & knowledge	Adequate	Inadequate	Inadequate
	Commitment	Adequate	Adequate	Adequate
Requirements	Changeability	(High, High)	(High, Normal)	(High, Normal)
	Standards	Inadequate	Adequate	Inadequate
Application	Degree of risk	High	High	High
	Complexity	High	Normal	High
	Size	Large	Normal	Large
	Connectivity	High	Normal	Normal
	Reuse	Normal	Normal	Normal
	Deployment profile	Normal	High	Normal
	Quality	High	High	High
Organization	Maturity	Adequate	Adequate	Adequate
	Management Commitment & expertise	(Adequate, Adequate)	(Adequate, Adequate)	(Adequate, Adequate)
	Facilities	Inadequate	Adequate	Inadequate
Operation	End-user experience	Inadequate	Adequate	Inadequate
Business	Time to market	Normal	Short	Normal
	External dependencies	High	High	Normal
	Opportunities	High	High	High
	Business drivers	Maximizing customer satisfaction	Financial considerations	Minimizing costs, Maximizing customer satisfaction
	Magnitude of potential loss	High	Normal	High

- If executing the transformations might result in processes that are not complete or syntactically correct; for example, if the dependencies among process elements were not correctly and completely identified in the metaprocess, then the produced process would have been useless.
- If executing the transformations might not result in a specific process since most of the variabilities remain unresolved based on the values of situational factors. This indicates that the set of situational factors, the range of values defined for each factor, or the situations defined for resolving the variabilities should be refined.

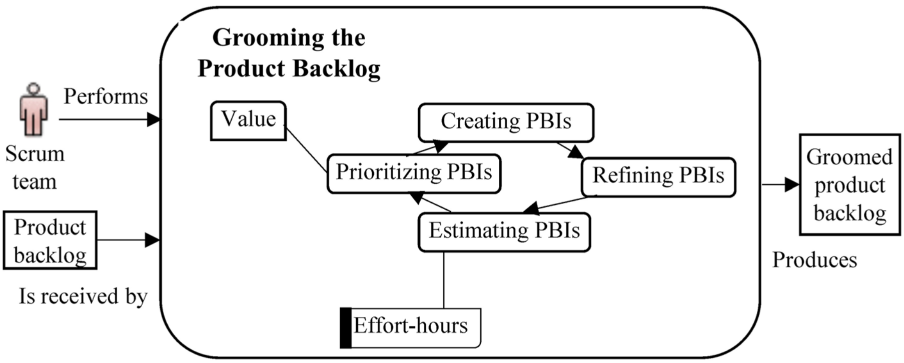

Fig. 18 "Grooming the product backlog" as currently used in project A.2

4.6.2 RQ2-Improvement of existing processes

To answer this question, we enquired the subjects about the feasibility of applying certain parts of the proposed processes in their upcoming sprints. The subject involved in project A.2 agreed to apply the following improvements: using the "story point" unit for estimating PBIs and the "effort hour" unit for estimating tasks, specifying a range of velocities for the development team, using the "planning poker" technique for estimating PBIs, specifying "definition of ready" criteria, conducting "competitive analysis", appointing the Scrum master as the facilitator for "sprint review" meetings, defining guidelines for presenting the demo in "sprint review" meetings, and separating "sprint review" and "sprint planning" sessions. Other subjects did not have enough time in their upcoming sprints for applying the proposed practices.

We designed a questionnaire for obtaining feedback on the impact of the practices applied in project A.2; this questionnaire is presented in Appendix A. The questionnaire was designed in two parts: in the first part, we designed specific questions for gathering details of the practitioners;

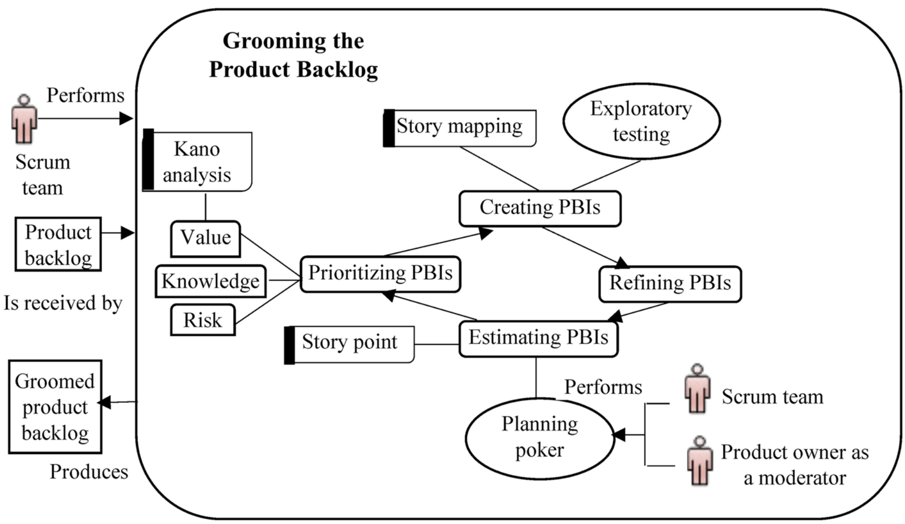

Fig. 19 Proposed "Grooming the product backlog" activity for project A.2

in the second part, the respondents were asked to give their opinions on whether the practices applied had resulted in improvements to the process used in project A.2. This questionnaire was filled out by the subject and development team members. The Likert response scale used in the questionnaire was as follows: strongly agree, agree, neutral, disagree, strongly disagree, and not sure. After applying the practices, we scheduled a meeting for gathering feedback from the subject and team members using the predesigned questionnaire. The duration of the meeting was 30 minutes. At the start of the meeting, we spent 10 minutes to explain the questionnaire. Subjects were then given 20 minutes to fill out the questionnaire. Figure 20 shows a summary of the responses given to these questions (provided by six respondents); as seen in this figure, almost all the respondents agreed that all the practices applied in project A.2 had had a positive effect on improving its process.

In addition to the questions about the practices applied, we also designed questionnaires for identifying the potential effects of some of the proposed practices on improving the processes currently used in projects A.1 and A.3. The questionnaires are provided in (Agh & Ramsin, 2019). Analyzing the responses (provided by eight respondents) to these questions indicated that the percentage of positive responses to the proposed practices was above 50%.

4.7 Discussion

The results of the case study confirm that the metaprocess is applicable to real situations (RQ1), as effective processes were custom-built for three running projects. The results of applying some of the improvements proposed show that the shortcomings of current processes can indeed be mitigated by the practices defined in the metaprocess (RQ2).

The proposed metaprocess provides significant advantages for practitioners who use Scrum for software development, and also for researchers of software process (re)engineering. Practitioners can use it at the start of a project to define an initial instance of Scrum, and also at retrospective meetings to refine/improve it. Identification of the variabilities, along

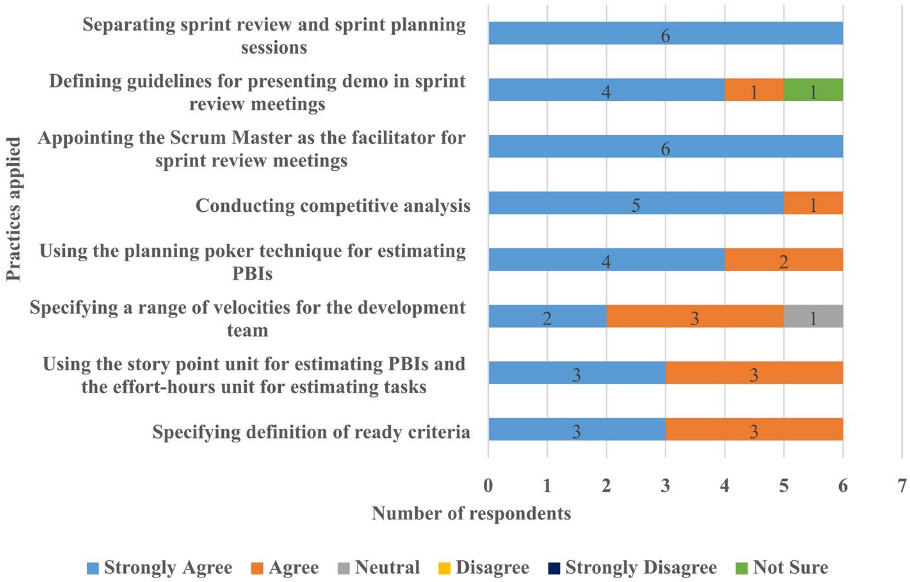

Fig. 20 Summary of responses in project A.2

with specific situations for their resolution, has been performed using existing resources on Scrum and agile methods (Padberg & Muller, 2003; Cohn, 2005; Ge et al., 2006; McDaid, 2006; Warden & Shore, 2007; Tingling & Saeed, 2007; Arisholm et al., 2007; Chong & Hurlbutt, 2007; O'Donnell & Richardson, 2008; Sterling, 2008; Maruping et al., 2009; Pichler, 2009; Cohn, 2010; Leffingwell, 2010; Causevic et al. 2011; Power, 2011; Rubin, 2012; Mahnič & Hovelja, 2012; Hajratwala, 2012; Greening, 2012; Heikkilä, 2013; Reddaiah et al., 2013; Shah et al., 2014; Power, 2014; Pauly et al., 2015; Jovanovic et al., 2015; Hussain & Javed, 2015; Stettina, 2015; McKenna, 2016; Savoine et al., 2016; Erdoğan et al., 2018). However, it is usually required that the situations defined for resolving the variabilities be adjusted based on the tailoring experience in the organization. Furthermore, the set of variants and situations defined for resolving the variabilities can be enhanced over time based on newly published works that report on the experiences of practitioners in using Scrum in different situations. In other words, the proposed metaprocess provides a good starting point for creating an initial instance of Scrum by using a systematic approach, but the metaprocess should be enhanced over time based on the experience gained from its use, and also based on the best practices shared by the agile community.

SME approaches have emerged to address the need for constructing software processes according to specific project/organizational situations (Henderson-Sellers et al., 2014). Systematic application of these approaches can help practitioners increase the quality of their software processes. Identifying the fixed and variable parts of an agile process, such as Scrum, and presenting them as a metaprocess provides a big-picture view of that process. There are three prominent, high-level SME approaches that can benefit from such a view, namely paradigm-based, assembly-based, and extension-based (Henderson-Sellers et al., 2014; Ralyté et al., 2003): if these SME approaches are fused into Scrum, method engineers can assist the practitioners in building or improving the target methodology by instantiating or adapting the metaprocess (paradigm-based SME), or by combining the different parts of the metaprocess with reusable parts retrieved from a repository (assembly-based SME), or by using the metaprocess as an extension framework to extend incomplete processes (extension-based SME). Research efforts in the fields of SME, SPI (Pino et al., 2008), and SPrLE (de Carvalho et al., 2014b) can benefit from the results of this research by abstracting, extending or reusing them to produce similar methods for different contexts.

4.7.1 Threats to validity

There are several threats to the validity of this study, including:

- **Internal validity**: One potential threat to internal validity is misinterpretations in the conversations made during interviews; to mitigate this threat, in addition to the notes taken during the interviews, voice recordings were produced, to later be transcribed as part of analysis. The results of each interview session were also reported back to the subjects via email and discussed in face-to-face conversations (*Member checking*). Subject fatigue was the other threat to internal validity, which was handled in our study by holding the interviews in multiple one-hour sessions.
- **Construct validity**: A potential threat to construct validity is subject selection. Random subject selection was not possible in our study since we needed subjects with adequate knowledge on the processes being used in the organization. However, the viewpoints of the different roles involved were considered throughout the case study as well as in the questionnaire designed for obtaining feedback on the proposed prac-

tices (*Theory triangulation*). Furthermore, a case study protocol was defined at the beginning of the study and was updated continuously afterwards (*Audit trail*).

- **Conclusion validity**: The reliability of the study results is one potential threat to conclusion validity. The techniques used to mitigate this risk are as follows: Reviewing the procedures selected for data collection and analysis by an expert (the second author), member checking, and theory triangulation.
- **External validity**: Threats to external validity are a serious issue in case studies. To mitigate this risk, we applied the approach on a real case. However, further evaluations are required to generalize the findings of this study. In particular, further industrial case studies should be performed to confirm that using the metaprocess in different situations can indeed result in the production of effective processes. Conducting such real case studies can help validate the variants defined in the metaprocess and the specific situations for resolving them, and can also enhance the metaprocess based on the experience gained.

5 Related research

The potential application scope of SPrLs is expanding (Golpayegani et al., 2013; Simmonds et al., 2015), but little has been done to provide SPrLs for prominent methodologies. In Ruiz and Hurtado (2012), a SPrL approach is presented for systematically adapting the Unified Process (UP); this work focuses on identifying the variabilities of the "Implementation" discipline of UP at the *task* level. In Ruiz and Hurtado (2012), the authors conclude that although defining SPrLs needs more effort than defining a unique process, the effort needed for defining specific processes is considerably reduced by using a SPrL. In de Carvalho et al. (2014a), a SPrL is proposed for integrating Scrum and CMMI; this work is limited to identifying the variabilities related to Project Planning and Project Monitoring and Control, without any attention to the relevant situations. In de Carvalho et al. (2014a), the authors conclude that the use of support tools for automating the various stages of process adaptation (e.g. automatic adaptation of the process according to the specifications of the project at hand) is considerably useful for adoption of SPrLs in real situations.

In contrast to the research works on providing SPrLs for prominent methodologies that lack adequate rigor and fail to provide enough detail on the variabilities and their resolution, we have identified a vast set of variabilities in the Scrum framework and have determined the situations where a variability can be resolved with specific variants. Furthermore, a certain level of automation has been provided in resolving the variabilities via transformations implemented in Medini-QVT.

6 Conclusion and future work

Software development methodologies are parameterized entities. However, their parameters (consisting of variabilities and applicable variants) are prohibitively numerous, even for agile methodologies. This research was conducted to address a clear and present challenge: the parameters of a software process cannot be set effectively unless all the variabilities, variants, relevant development situations, and their interrelationships, are properly identified. Scrum was targeted because of its importance as the most popular agile process, and also because of its framework nature and rich set of variants: since it is presented as a high-level framework, its parameterized nature has been deliberately accentuated, and over more than two decades, a myriad of variants (activities, roles, and products) have been developed or adapted for use in Scrum. The metaprocess that we propose has been complemented with situational factors that express the different situations that may arise in agile contexts, as well as the variants that resolve each variability in each situation. This provides the knowledge required for applying the methodology to the benefit of Scrum team(s) and stakeholders. Moreover, the proposed approach is of potential research value in SME, SPI and SPrLE.

The proposed metaprocess has been assessed by conducting an industrial case study; the results of which confirm the applicability of the metaprocess in real situations to produce specific Scrum processes. Furthermore, the produced processes can improve existing processes by proposing best practices for addressing their shortcomings.

This research can be furthered in several directions, including: enriching the set of variants by reusing the activities, roles and products used in other agile/non-agile methodologies, refining the situations defined for resolving the variabilities by conducting further case studies, and identifying other factors affecting agile processes.

Appendix: Questionnaire designed for obtaining feedback on the practices applied

The questionnaire was designed in two parts, described in Sect. 7.1 and Sect. 7.2, respectively.

Part one

This part focused on gathering practitioner details. The specific questions designed for this purpose are shown in Table 15.

Table 15 Questions on practitioner details

Practitioner detail	
Full name (optional)	
Job title/position	
Telephone No. (optional)	
Email address	
How many years of industrial experience do you have in your field?	☐ Less than 1 year ☐ 1–2 years ☐ 3–5 years ☐ 6–10 years ☐ More than 10 years
How many software development projects have you been involved in?	☐ 1–2 ☐ 3–4 ☐ 5–7 ☐ 8–10 ☐ Above 10
Have you ever taken part in a project in which it was necessary to adapt/tailor the process/methodology prior to or during application?	☐ Yes ☐ No
If your previous answer was Yes, please rate the grade of tailoring/adaptation applied to the methodology/process	☐ High degree ☐ Moderate degree ☐ Small degree

Part two

In this part, participants were asked to state their opinion on whether each of the practices listed in Table 16 would result in improving the process being used in their project.

Table 16 Questions designed for obtaining feedback on the practices applied

	Strongly agree	Agree	Neutral	Disagree	Strongly disagree	Not sure
Specifying the "Definition of Ready" criteria	☐	☐	☐	☐	☐	☐
Using the story-point unit for estimating PBIs and the effort-hours unit for estimating tasks	☐	☐	☐	☐	☐	☐
Specifying a range of velocities for the development team	☐	☐	☐	☐	☐	☐
Using the "Planning Poker" technique for estimating PBIs	☐	☐	☐	☐	☐	☐
Conducting competitive analysis	☐	☐	☐	☐	☐	☐
Appointing the Scrum master as the facilitator for sprint review meetings	☐	☐	☐	☐	☐	☐
Defining guidelines for presenting the demo in sprint review meetings	☐	☐	☐	☐	☐	☐
Separating sprint review and sprint planning sessions	☐	☐	☐	☐	☐	☐

References

Abad, Z. S. H., Alipour, A., & Ramsin, R. (2012). Enhancing tool support for situational engineering of agile methodologies in Eclipse. In R. Lee (Ed.), *Software Engineering Research, Management and Applications* (pp. 141–152).

Agh, H., & Ramsin, R. (2019). Scrum Metaprocess: A Process Line Approach for Customizing Scrum-Appendices. Mendeley Data. https://doi.org/10.17632/f5dbbsj3p6.1

Ahmad, M. O., Kuvaja, P., Oivo, M., & Markkula, J. (2016). Transition of software maintenance teams from Scrum to Kanban. *Proc. Hawaii International Conference on System Sciences*, 5427–5436.

Aleixo, F.A., Freire, M.A., dos Santos, W.C., & Kulesza, U. (2010). Automating the variability management, customization and deployment of software processes: A model-driven approach. *Proc. International Conference on Enterprise Information Systems*, 372–387.

Ally, M., Darroch, F., & Toleman, M., et al. (2005). A framework for understanding the factors influencing pair programming success. In H. Baumeister (Ed.), *Extreme Programming and Agile Processes in Software Engineering* (pp. 82–91).

Arisholm, E., Gallis, H., Dyba, T., & Sjoberg, D. I. (2007). Evaluating pair programming with respect to system complexity and programmer expertise. *IEEE Transactions on Software Engineering, 33*(2), 65–86.

Campanelli, A. S., & Parreiras, F. S. (2015). Agile methods tailoring–A systematic literature review. *Journal of Systems and Software, 110,* 85–100.

Canty, D. (2015). *Agile for project managers*. Auerbach Publications.

Causevic, A., Sundmark, D., & Punnekkat, S. (2011). Factors limiting industrial adoption of test driven development: A systematic review. *Proc. IEEE International Conference on Software Testing, Verification and Validation*, 337–346.

Chong, J., & Hurlbutt, T. (2007). The social dynamics of pair programming. *Proc. International Conference on Software Engineering*, 354–363.

Clarke, P., & O'Connor, R. V. (2012). The situational factors that affect the software development process: Towards a comprehensive reference framework. *Information and Software Technology, 54*(5), 433–447.

Cohn, M. (2005). *Agile estimating and planning*. Pearson Education.

Cohn, M. (2010). *Succeeding with agile: Software development using Scrum*. Pearson Education.

Cram, W. A. (2019). Agile development in practice: Lessons from the trenches. *Information Systems Management, 36*(1), 2–14.

de Carvalho, D.D., Chagas, L.F., & Reis, C.A.L. (2014a). Definition of software process lines for integration of Scrum and CMMI. *Proc. XL Latin American Computing Conference*, 1–12.

de Carvalho, D. D., Chagas, L. F., Lima, A. M., & Reis, C. A. L., et al. (2014b). Software process lines: A systematic literature review. In A. Mitasiunas (Ed.), *Software Process Improvement and Capability Determination* (pp. 118–130).

Diebold, P., Ostberg, J.P., Wagner, S., & Zendler, U. (2015). What do practitioners vary in using Scrum? *Proc. International Conference on Agile Software Development*, 40–51.

Erdoğan, O., Pekkaya, M. E., & Gök, H. (2018). More effective Sprint Retrospective with statistical analysis. *Journal of Software: Evolution and Process, 30*(5), e1933.

Fitzgerald, B., Hartnett, G., & Conboy, K. (2006). Customizing agile methods to software practices at Intel Shannon. *European Journal of Information Systems, 15*(2), 200–213.

Ge, X., Paige, R.F., Polack, F.A., Chivers, H., & Brooke, P.J. (2006). Agile development of secure web applications. *Proc. International Conference on Web engineering*, 305–312.

Gill, A.Q., & Henderson-Sellers, B. (2006). Measuring agility and adoptability of agile methods: A 4-dimensional analytical tool. *Proc. IADIS International Conference on Applied Computing*, 503–507.

Golpayegani, F., Azadbakht, K., & Ramsin, R. (2013). Towards process lines for agent-oriented requirements engineering. *Proc. International Conference on Computer as a Tool*, 550–557.

Greening, D. (2012). Bulk Estimation. https://senexrex.com/bulk-estimation/ Accessed 16 November 2019.

Hajratwala, N. (2012). Task board evolution. *Proc. International Agile Conference*, 111–116.

Heeager, L. T., & Rose, J. (2015). Optimising agile development practices for the maintenance operation: Nine heuristics. *Empirical Software Engineering, 20*(6), 1762–1784.

Heikkilä, V.T., Paasivaara, M., Lassenius, C., & Engblom, C. (2013). Continuous Release Planning in a large-scale Scrum development organization at Eriksson. *Proc. International Conference on Agile Software Development*, 195–209.

Henderson-Sellers, B., Ralyté, J., Agerfalk, P. J., & Rossi, M. (2014). *Situational method engineering*. Springer.

Hoda, R., Kruchten, P., Noble, J., & Marshall, S. (2010). Agility in context. *Proc. ACM International Conference on Object Oriented Programming, Systems, Languages and Applications*, 74–88.

Hodgetts, P. (2004). Refactoring the development process: Experiences with the incremental adoption of agile practices. *Proc. IEEE Agile Development Conference*, 106–113.

Hurtado, J. A., Bastarrica, M. C., Ochoa, S. F., & Simmonds, J. (2013). MDE software process lines in small companies. *Journal of Systems and Software, 86*(5), 1153–1171.

Hussain, R. G., & Javed, A. (2015). Qualitative approach for estimating the influence of refactoring and Scrum in Software Development. *International Journal of Engineering Research and General Science, 3*(2), 28–36.

Ibrahim, K. S. K., Yahaya, J., Mansor, Z., & Deraman, A. (2019). The Emergence of agile Maintenance: A preliminary study. *Proc. International Conference on Electrical Engineering and Informatics*, 146–151.

Jovanovic, M., Mesquida, A.L., & Mas, A. (2015). Process improvement with retrospective gaming in agile software development. *Proc. European Conference on Software Process Improvement*, 287–294.

Junior, E. A. O., Pazin, M. G., Gimenes, I. M., Kulesza, U., & Aleixo, F. A., et al. (2013). SMartySPEM: A SPEM-Based Approach for Variability Management in Software Process Lines. In J. Heidrich (Ed.), *Product-Focused Software Process Improvement* (pp. 169–183).

Jyothi, V. E., & Rao, K. N. (2012). Effective implementation of agile practices: In coordination with lean Kanban. *International Journal on Computer Science and Engineering, 4*(1), 87–91.

Karlsson, F., & Ågerfalk, P. (2009). Exploring agile values in method configuration. *European Journal of Information Systems, 18*(4), 300–316.

Kaur, C., & Kumar, V. (2015). Product backlog prioritization in Scrum: A review. *International Journal of Modern Computer Science, 3*(2), 21–24.

Kniberg, H., & Skarin, M. (2010). *Kanban and Scrum - Making the most of both*. Lulu.com.

Kruchten, P. (2013). Contextualizing agile software development. *Journal of Software: Evolution and Process, 25*(4), 351–361.

Larman, C. (2008). *Scaling lean & agile development: Thinking and organizational tools for large-scale Scrum*. Pearson Education.

Law, A., & Charron, R. (2005). Effects of agile practices on social factors. *Proc. International Workshop on Human and Social Factors of Software Engineering*, 1–5.

Leffingwell, D. (2010). *Agile software requirements: Lean requirements practices for teams, programs, and the enterprise*. Addison-Wesley Professional.

Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., Shull, F., et al. (2002). Empirical findings in agile methods. *Proc. International Conference on extreme programming and agile methods*, 197–207.

Mahnič, V., & Hovelja, T. (2012). On using planning poker for estimating user stories. *Journal of Systems and Software, 85*(9), 2086–2095.

Martínez-Ruiz, T., García, F., Piattini, M., & Munch, J. (2011). Modelling software process variability: An empirical study. *IET Software, 5,* 172–187.

Maruping, L. M., Zhang, X., & Venkatesh, V. (2009). Role of collective ownership and coding standards in coordinating expertise in software project teams. *European Journal of Information Systems, 18*(4), 355–371.

Masood, Z., Hoda, R., & Blincoe, K. (2020). Real World Scrum A Grounded Theory of Variations in Practice. *IEEE Transactions on Software Engineering*, 1–13.

McDaid, K., Greer, D., Keenan, F., Prior, P., Coleman, G., & Taylor, P.S. (2006). Managing Uncertainty in Agile Release Planning. *Proc. International Conference on Software Engineering and Knowledge Engineering*, 138–143.

McKenna, D. (2016). *The Art of Scrum: How Scrum Masters Bind Dev Teams and Unleash Agility*. Apress.

Medini QVT: IKV + + technologies home. http://projects.ikv.de/qvt Accessed 16 November 2019.

O'Donnell, M.J., & Richardson, I. (2008). Problems encountered when implementing agile methods in a very small company. In R.V. O'Connor et al. (Eds.), *Software Process Improvement*, 13–24.

OMG: Software & Systems Process Engineering Metamodel Specification Version 2.0. (2008). https://www.omg.org/spec/SPEM/About-SPEM/ Accessed 16 November 2019.

Paasivaara, M., Lassenius, C., & Heikkilä, V.T. (2012). Inter-team coordination in large-scale globally distributed Scrum: Do Scrum-of-Scrums really work? *Proc. ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 235–238.

Padberg, F., & Muller, M.M. (2003). Analyzing the cost and benefit of pair programming. *Proc. IEEE International Symposium on Software Metrics*, 166–177.

Palomino, M., Dávila, A., Melendez, K. & Pessoa, M. (2016). Agile practices adoption in CMMI organizations: A systematic literature review. *Proc. International Conference on Software Process Improvement*, 57–67.

Pato, R. H., Granada, D., Vara, J. M., & Marcos, E. (2020). Lean Kanban in an industrial context: A success story. *Proc. ACM/IEEE International Conference on Software Engineering*, 282–283.

Pauly, D., Michalik, B., & Basten, D. (2015). Do daily Scrums have to take place each day? A case study of customized Scrum principles at an e-commerce company. *Proc. Hawaii International Conference on System Sciences*, 5074–5083.

Pichler, R. (2009). The Product Vision. Available online at https://www.scrumalliance.org/community/articles/2009/january/the-product-vision

Pino, F. J., García, F., & Piattini, M. (2008). Software process improvement in small and medium software enterprises: A systematic review. *Software Quality Journal, 16*(2), 237–261.

Power, K. (2011). Using silent grouping to size user stories. *Proc. International Conference on Agile Software Development*, 60–72.

Power, K. (2014). Metrics for Understanding Flow. *Proc. International Conference on Agile Software Development*, 1–8.

Proba, D., & Jung, R. (2019). Defining Situational Characteristics for Situational Agile Method Engineering, Defining Situational Characteristics for Situational Agile Method Engineering. *Proc. Americas Conference on Information Systems*, 1–10.

Ralyté, J., Deneckère, R., & Rolland, C. (2003). Towards a generic model for situational method engineering. *Proc. International Conference on Advanced Information Systems Engineering*, 95–110.

Reddaiah, B., Ravi, S. P., & Movva, L. S. (2013). Risk management board for effective risk management in Scrum. *International Journal of Computer Applications, 65*(12), 16–23.

Reinertsen G.D. (2009). Types of Processes. Available online at http://www.netobjectives.com/blogs/Types-of-Processes

Rubin, K.S. (2012). *Essential Scrum: A practical guide to the most popular agile process*. Addison-Wesley.

Ruiz, P.H., & Hurtado, J.A. (2012). A software process line based on the Unified Process. *Proc. Colombian Computing Congress*, 1–6.

Runeson, P., Host, M., Rainer, A., & Regnell, B. (2012). *Case study research in software engineering: Guidelines and examples*. John Wiley & Sons.

Savoine, M.M., Rocha, V.F., Bezerra, C.A.C., de Araújo, A.M.C., & Matias, J.K.M. (2016). A Synchronous Agile Framework Proposal Combining Scrum and TDD. *Proc. International Conference on Software Engineering Advances*, 337–341.
Schwaber, K. (2015). The NexusTM Guide. https://www.scrum.org/resources/nexus-guide Accessed 16 November 2019.
Schwaber, K., & Sutherland, J. (2017). Scrum Guide. https://www.scrum.org/resources/scrum-guide Accessed 16 November 2019.
Seikola, M., & Loisa, H. M. (2011). Kanban implementation in a telecom product maintenance. *Proc. EUROMICRO Conference on Software Engineering and Advanced Applications*, 321–329.
Shah, S. M. A., Torchiano, M., Vetro, A., & Morisio, M. (2014). Exploratory testing as a source of technical debt. *IT Professional, 16*(3), 44–51.
Simmonds, J., Perovich, D., Bastarrica, M. C., & Silvestre, L. (2015). A megamodel for Software Process Line modeling and evolution. *Proc. ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, 406–415.
Sjøberg, D. I., Johnsen, A., & Solberg, J. (2012). Quantifying the effect of using Kanban versus Scrum: A case study. *IEEE software, 29*(5), 47–53.
Stankovic, D., Nikolic, V., Djordjevic, M., & Cao, D. B. (2013). A survey study of critical success factors in agile software projects in former Yugoslavia IT companies. *Journal of Systems and Software, 86*(6), 1663–1678.
Sterling, C. (2008). Affinity Estimating: A How-To. http://www.gettingagile.com/2008/07/04/affinity-estimating-a-how-to/ Accessed 16 November 2019.
Stettina, C. J., & Hörz, J. (2015). Agile portfolio management: An empirical perspective on the practice in use. *International Journal of Project Management, 33*(1), 140–152.
Tingling, P., & Saeed, A. (2007). Extreme Programming in Action: A Longitudinal Case Study. In J. A. Jacko (Ed.), *Human-Computer Interaction* (pp. 242–251).
Uikey, N., & Suman, U. (2016). Tailoring for agile methodologies: A framework for sustaining quality and productivity. *International Journal of Business Information Systems, 23*(4), 432–455.
Warden, S., & Shore, J. (2007). *The art of agile development*. O'Reilly.
Yi, L. (2011). Manager as Scrum Master. *Proc. International Agile Conference*, 151–153.

Halimeh Agh is a PhD candidate at Sharif University of Technology, working on Software Process Line Engineering (SPrLE). She received her MSc degree in Computer Engineering (Software) from Sharif University of Technology in 2013. Her research interests include situational method engineering, model-driven development, and software processes.

Raman Ramsin is an Assistant Professor at the Department of Computer Engineering, Sharif University of Technology. He received his PhD in Computer Science from the University of York, UK, in 2006. His research focuses on model-driven engineering, situational method engineering, and agile software development.