Information Technology and Management
https://doi.org/10.1007/s10799-023-00386-y

A knowledge management-driven and DevOps-based method for situational method engineering

Razieh Dehghani[1] · Raman Ramsin[1]

Accepted: 28 December 2022

Abstract
Earlier software development processes (SDPs), such as waterfall processes, were mainly focused on process steps and did not address people- and product-related issues. Emergence of Software development methodologies (SDM) has created a new paradigm for developing software systems. A SDM is a special kind of technically engineered framework for organizing SDPs; this framework is expected to specify three main interwoven elements, namely people, products, and process. It has since become evident that it is impossible to provide a general-purpose SDM for developing all the various kinds of software systems, and it has thus become essential to construct the most appropriate methodology for the system development situation in hand, a practice commonly called Situational Method Engineering (SME). The problem with existing SME methods is lack of adequate attention to the role of people who might seek or possess valuable knowledge about the project situation. This knowledge can be tacit information that is hidden in the developer's mind, or it might be explicitly available. This paper proposes a knowledge management (KM)-driven and DevOps-based SME method as a new integrated multi-view methodological paradigm that satisfies the need for sharing human experience in engineering SDMs. The method has been proposed by reusing general SME practices and complementing them by embedding appropriate KM and DevOps practices to alleviate the weaknesses of previous SME methods. Furthermore, the proposed method has been evaluated through four case studies and also by conducting a criteria-based comparison with eight prominent SME methods.

Keywords Software process engineering · Knowledge management-driven improvement · Software process knowledge · DevOps-based process · Situational method engineering · Developer experience management

1 Introduction

This work draws attention to an important, yet neglected, concern: using developers' experience for improving the quality of Software Development Processes (SDPs). Developers can share their opinions regarding the processes they use, even though they are not experts in process engineering. We have observed that, in comparison with method engineers, developers can develop a deeper understanding of the differences between the characteristics of various projects, and thus, the various SDPs that should be applied.

✉ Raman Ramsin
ramsin@sharif.edu

Razieh Dehghani
rdehghani@ce.sharif.edu

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

Primitive SDPs were mainly focused on process steps. The emergence of Software Development Methodologies (SDMs) has created a new perspective to SDPs. A SDM has been defined as a well-integrated framework of appropriate activities, roles, techniques, products, and guidelines for engineering software systems [1]. It is impossible to provide a general-purpose SDM for developing all the various kinds of software systems, and it has thus become essential to construct the most appropriate methodology for the system development situation at hand, a practice commonly called Situational Method Engineering (SME). A "situation" is described through a set of specific characteristics. As an example, different projects and organizations have different characteristics, and prominent SDMs such as Scrum are used differently in each of these situations.

SME can be seen as a knowledge-seeking effort to engineer (build or customize) a SDM in accordance with the features of the project at hand [2]. Existing SME methods have focused on the process aspect of engineering, and proposed four main steps which respectively pursue the following goals: (1) deciding about engineering a new SDM or improving an existing one, (2) specifying the most appropriate strategy for constructing the target SDM, (3) evaluating constructed method parts, and (4) ensuring the completeness of engineered SDM [2]. So far, five main high-level approaches have been proposed for constructing SDMs, namely "Assembly-Based", "Paradigm-Based", "Deontic Matrix-Based", "Activity Diagram-Based", and "Configuration-Based" [2].

The important problem with these approaches is the lack of adequate practices for engineering concrete *reusable* and *practicable* SDMs, instead of producing high-level well-documented instructions [2]. Thus, the applicability of two categories of practices in SME processes has been studied in this research: (1) Knowledge Management (KM) practices that help share the developers' valuable experiences, and (2) Development and Operations (DevOps) practices that support the engineering and maintenance of more operationalized SDMs. Taking advantage of these practices to propose a new KM-driven and DevOps-based SME method is the main novelty of this research. To explain the results, it is first required to define the notions of KM and DevOps in SME; the following two paragraphs provide these definitions.

SME knowledge refers to the "State of Mind", "Object", and "Process" dimensions of the knowledge hidden in method engineering operations [3]. These dimensions are proposed based on the three main constituents of a SDM, namely "People", "Product", and "Process" [2]. On this basis, managing SME knowledge refers to creating an appropriate learning environment for method engineers and developers (people), building and managing valuable knowledge assets (including products), and facilitating the flow of knowledge throughout method engineering and thus software development activities (process); this should encompass managing both tacit and explicit types of knowledge. The tacit type, which might be hidden in processes, products, or individuals' minds, is more difficult to share. Hence, our proposed method takes advantage of several KM practices to alleviate the weaknesses of existing SME methods in establishing the flow of both tacit and explicit types of SME knowledge.

Henderson-Sellers et al. have referred to several knowledge sub-areas in SME [2], and we have categorized these areas in three classes (Table 1). DevOps practices help develop and operationalize software systems [4], and some DevOps practices facilitate sharing the knowledge required for operationalization [5]. Since "software processes are

Table 1 Synergistic relationships between SME knowledge areas and DevOps practices

SME knowledge area	SME knowledge sub-areas (Adapted from [2])	Corresponding DevOps practices (Adapted from [4])	Common concerns
Management	Structuring SDM Constituents (Knowledge Base)	Process Standardization	Specifying a unique and reusable method for presenting and updating process constituents
	Human Resource Management	Feedback Loop	Supporting iterative-incremental learning and improvement
	Representational Approach	Prototyping	Receiving early feedback and avoidance of late discovery of misunderstandings
	Constituent Integration	Continuous Release	Delivering resilient products and ensuring early discovery of problems
	Project Management	Continuous Planning	Making realistic plans
	Quality Assurance	Continuous Monitoring	Discovering quality threats as soon as possible
Process	SDM Enacting	Continuous Revision	Delivering practicable products
	Tailoring and Configuration	Change Management	Preserving the logic behind changes and support for tailoring products by rolling back the changes
	Construction Strategy	Continuous Building	Coordination between building strategy and constructed products
	Tool Usage	Continuous Preparation	Coordination between execution platform and running processes
	SDM Evolution	Continuous Delivery	Making release processes repeatable
	Situation Analysis	Defining Requirements	Developing specific-purpose products
	SDM Evaluation	Continuous Testing	Learning about products continuously and improving reliability
Modeling	Meta-Modeling	"Modeling and Simulation"	Providing an appropriate platform for shared understanding

software too" [6], our proposed method uses these practices for operationalizing SME methods, which in turn produce operationalized SDMs. Table 1 shows the relationship between DevOps practices and SME knowledge areas. It should be noted that, depending on the goals pursued, other kinds of relations might also be proposed. We have found these relations by studying the requirements in managing SME knowledge and the corresponding applications of DevOps practices. Thus, the correspondence provided in Table 1 has been proposed by focusing on the issues discussed by Henderson-Sellers et al. [2] and Jabbari et al. [4]. It should be noted that KM sub-areas are not the same as DevOps practices, but we have observed that they address certain common concerns (as shown in Table 1), based on which they can be combined to provide synergistic results.

The remaining sections will focus on the following issues: the related work; the method used for deriving the constituents of our proposed method; the proposed method itself; results of evaluating the proposed method; and conclusions and future work.

2 Related work

This research aims to help fill the following knowledge gaps (inspired by the risks introduced in [7]) with available SME methods: (1) "knowledge attrition": The SDPs or the products, produced throughout the SDP, become obsolete. Besides, they might be used inappropriately; (2) "knowledge loss": Individuals' (software developers'/method engineers'/project managers') knowledge or the products are lost; (3) "knowledge leakage": The right of ownership is violated for the products and SDPs; (4) "knowledge spillover": Those products, people, or SDPs that create competitive advantage are not preserved; (5) "lost reputation": The SDM seems impractical due to negative points of view about individuals' skills, quality of products, and practicality of SDPs; (6) "lost sustainability": The SDM is not flexible enough, and it is difficult to configure it to satisfy the new development requirements.

To reach the target mentioned above, the method proposed in this paper facilitates process and product improvement by prescribing KM-supportive mechanisms for concentrating on situational needs and avoiding distractive issues. We have thus studied three categories of work related to this paper, as explained in the following subsections.

2.1 Available SME methods

Existing SME methods have not directly addressed the management of SME knowledge, though they have used certain techniques for knowledge sharing. In Sect. 5, we will compare our proposed method with eight of these methods, which provide a more comprehensive support for KM process. These methods have been selected based on the availability and adequacy (richness in detail) of their support documentation. Furthermore, we have evaluated the available SME methods by using a set of criteria that we have previously elicited to evaluate the support that SME methods provide for managing SME knowledge [8]. In addition, we have used the framework provided in [9] for comparing and selecting superior methods based on the following features: nature, construction technique, knowledge representation technique, supported dimension (process/product), abstraction approach, support for formalism, support for flexibility, knowledge construction approach, and knowledge structuring method.

Table 2 provides a general overview of seven of the methods selected. This schema has been provided by specifying the correspondence between a general SME process and features of the SME methods selected. The general process has been introduced in [2], and consists of four high-level activities: Specifying principal approach for SDM construction, Selecting construction strategy, Evaluating method parts, and Validating the produced SDM as to completeness.

The eighth method has been inspired by the general SME process, and is superior to the other seven methods. We will thus describe it in more detail. Henderson-Sellers et al. have listed the following seven approaches for constructing a SDM: Assembly-Based, Extension-Based, Deontic Metrics, Activity Diagrams, Ad Hoc, Configuration-Based, and Paradigm-Based [2]. By comparing these approaches, it has been revealed that in comparison with other approaches, the assembly-based approach is more powerful in establishing the required knowledge flows [8]. Therefore, we have selected a special instance of the assembly-based approach (introduced in [17]) as the eighth SME method; this method provides a requirements-based process for selecting and assembling the appropriate method chunks [17].

The requirements are elicited by considering the project characteristics, which are categorized into four dimensions, namely: Organizational, Human, Application Domain, and Development Strategy [2]. As shown in Fig. 1, this method prescribes a five-step process:

a) *Specifying Project Characteristics* Investigate the project characteristics within the four categories provided.
b) *Specifying SDM Requirements* Specify the requirements through asking development and management team members to participate in requirements elicitation activities.
c) *Selecting Method Chunks* Select the appropriate chunks from the method chunk repository based on their weights and priorities.
d) *Assembling Method Chunks* Assemble the selected chunks to build the target SDM.

Table 2 General overview of available SME methods

General SME activity	Corresponding activity in evaluated SME method	Assigned role	Products	
			Input	Output
Hybrid Method [10]				
{1}	Selecting the design approach	NME	Process-Centered Descriptions of Available SDMs, Results of Analyzing Available SDMs	Method Parts
{2}	Selecting the design approach	NME	(Same as above)	(Same as above)
{3}	Revising the designed method	NME	Designed SDM	Designed SDM
{4}	Finalizing the designed method	NME	Designed SDM	Finalized SDM
Assembly-Based and Roadmap-Driven Method [11]				
{1}	Specifying method requirements	Method Engineer	Project Status, Reuse Framework, Requirements Specification Guidelines	Requirements Map, Reuse Framework
{2}	Specifying strategies of requirements satisfaction, Selecting method chunks	Method Engineer	Repository (of Method Chunks), Requirements Map, Guidelines for Chunk Selection	Repository
{3}	Retrieving method chunks	ISD Crew Member	Problem Context	Repository
{4}	Completing target method	ISD Crew Member	Repository	Repository, Final SDM
Domain-Driven Method [12]				
{1}	Specifying corresponding domain elements	Method Engineer	Domain Layer Elements	Selected Domain Layer Elements
{2}	Specifying corresponding domain elements, Investigating structural and behavioural aspects of method components, Investigating relationships between components	Method Engineer	Domain Layer Elements	Pre-Conditions and Post-Conditions, Selected Method Components
{3}	Retrieving and tailoring method components, Removing compositions that violate the composition-related tailoring info, Maintaining consistency of diagrams by the provided tool	Method Engineer	Method Components	Components of the SDM
{4}	NC	N/A	N/A	N/A
Agile OPEN Framework-Based Method [13]				
{1}	Adopting predefined examples	SME Team	Examples Provided by OPEN Guidelines	Adopted SDM
{2}	Conducting brainstorming sessions	SME Team, IT Personnel, Customers	Adopted SDM	Adopted SDM
{3}	Conducting brainstorming sessions	SME Team, IT Personnel, Customers	Adopted SDM	Adopted SDM
{4}	Checking all concerns and needs iteratively	SME Team, IT Personnel, Customers	Adopted SDM	Finalized SDM

Table 2 (continued)

General SME activity	Corresponding activity in evaluated SME method	Assigned role	Products	
			Input	Output
Goal-Oriented Method [14]				
{1}	Process analysis, Process development	NME	Organizational Maturity Level, Problem Types, Available Organizational Development Experience, Meta-Model for PASSI and Tropos, Evaluation Results	Process Capabilities and Requirements, Meta-Model Concepts, Inconsistencies
{2}	Process design	NME	Process Requirements, Inconsistencies, Method Base	Method Base, Goal-Oriented Meta-Model
{3}	Process development	NME	Goal-Oriented Meta-Model, Method Base, CASE Tools, CMMI Model	Evaluation Results
{4}	Process development	NME	(Same as above)	(Same as above)
Practice-Driven Method [15]				
{1}	Developing a meta-model for the organization's base SDM, Evolving the base SDM	Method Engineer	Base SDM, Current Practice (Informal SDM), Updated SDM	Base SDM, Updated SDM, Meta-Model
{2}	Specifying construction rules, Evolving the base SDM, Configuring the process	Method Engineer	Base SDM	Base SDM, Construction Rules, Project-Specific SDM
{3}	Assessing project performance	Method Engineer	Project-Specific SDM, Current Practice	Project Performance
{4}	Assessing project performance	Method Engineer	Project-Specific SDM, Current Practice	Project Performance
Game Design-Driven Method [16]				
{1}	Specifying game representations (method modeling) rules	Game Designer	Method Representation Rules	Game Representation Rules
{2}	Specifying game representations (method modeling) rules	Game Designer	Method Representation Rules	Game Representation Rules
{3}	Assessing utility of the game by modeling emotive factors as displayable game elements	Game Designer	Designed Game (SDM)	Emotive Game Elements
{4}	NC	N/A	N/A	N/A

{1}: Specifying principal approach for SDM construction;

{2}: Selecting construction strategy;

{3}: Evaluating method parts;

{4}: Validating completeness of SDM

NME Not Mentioned Explicitly (might be implicit), *NC* Not Covered, *N/A* Not Applicable

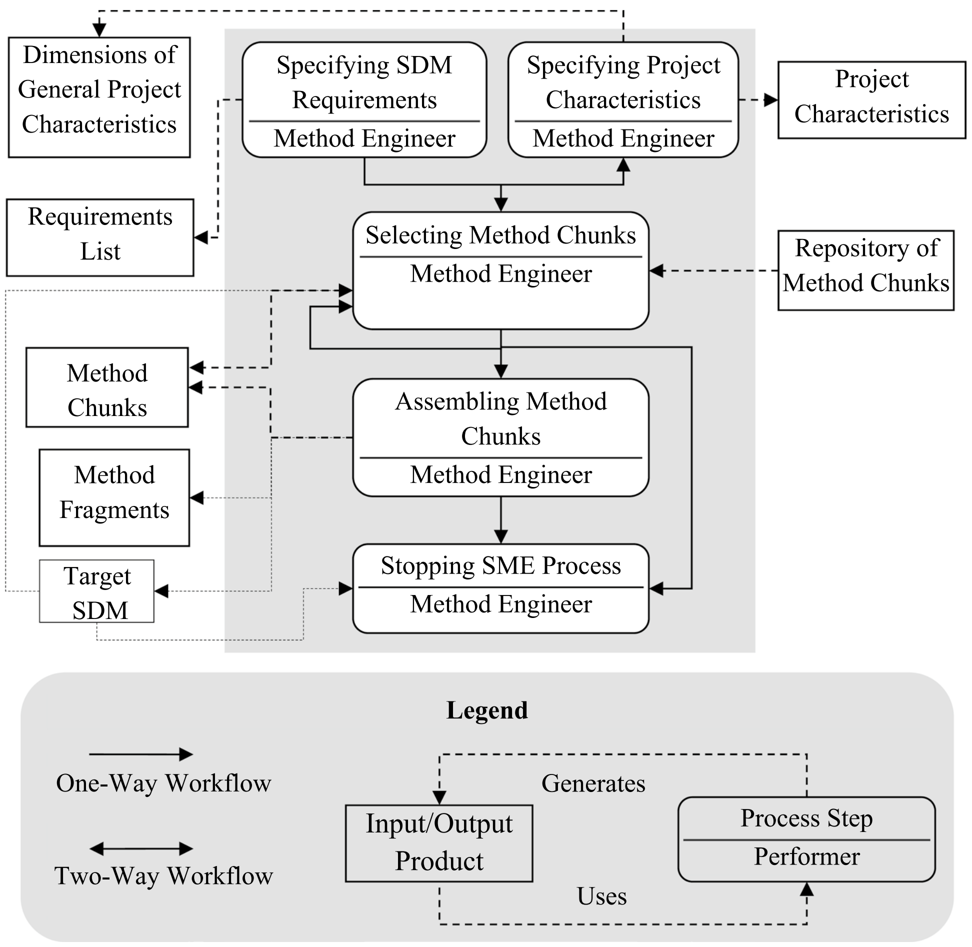

Fig. 1 SME method inspired by the assembly-based approach [17]

e) *Stopping SME process* Stop the SME process if the method satisfies the requirements.

2.2 Recent SME applications

Recent applications of SME have customized SME mechanisms for specific purposes. We have categorized the customization techniques as follows:

1. Abstraction: Some researchers have applied abstraction to available intention-based development practices, and proposed reusable customizable contributions. Karagiannis et al., for example, have used this technique to analyze the modeling method requirements. They have proposed high-level concepts for conceptualizing method requirements [18]. In fact, they have reused SME mechanisms for situational method requirements engineering.
2. Pattern Extraction: Many efforts have been made to identify software development patterns. These patterns provide proven solutions for recurrent software development problems in specific contexts. Some researchers have mapped these solutions to general, customizable practices. For example, Janković et al. have analyzed data repositories of projects conducted in specific companies, and elicited the methodology constituents and the relationships between them [19]. In fact, they have used a bottom-up approach to find reusable methodology constituents by generalizing the recurrent solutions extracted from data repositories of specific projects.
3. Inspiration: The research in this category has been inspired by SME practices. For example, Sandkuhl and Seigerroth have proposed a balanced scorecard approach for method improvement. This method takes advantage of a SME process to generate and enact scorecards [20].
4. Tailoring: These works use SME to tailor a solution to a specific context. Franch et al. have used SME for tailoring a data-driven software evolution method [21]. For this purpose, they have analyzed the elements of the evolution method, and extracted a metamodel for method chunks. Finally, they have proposed a three-step process for tailoring and assembling the method chunks to build a customized software evolution method.

2.3 Available knowledge sharing and quality improvement approaches

An important subset of the research related to this paper focuses on finding knowledge-sharing facilitators and obstacles. The following have been addressed by researchers in this context:

- Technical facilitators help preserve the logic underlying the engineered SME method and elicit tacit SME knowledge; ontologies [2] and games [16] are examples of such facilitators. Some researchers have built technological platforms to preserve and share situational knowledge. For example, Mishra et al. have created an online tool to share situational requirements engineering knowledge [22]. SME methods could provide guidance on creating and using such tools throughout the SDP.
- Trust and cultural issues affect the method engineers' mindset, and facilitate the acquisition of situational knowledge [23] [24]. For instance, Lopez et al. have proposed a situational method to help adopt open source software; however, they have explained that cultural issues affect the usability of their method [25].
- Obstacle analysis methods help improve SME knowledge flow by avoiding the obstacles recognized. For example, Mitchell and Seaman have used k-mapping to find obstacles in knowledge flows [26]. Such methods could be reused by SME methods to find and avoid knowledge flow obstacles continuously.

Another subset of the related research focuses on using Quality Improvement (QI) practices. SME methods should improve the quality of in-use SDMs. Thus, techniques for improving the quality of SDPs [27] are potentially useful in SME methods; specifically, a combination of KM and QI practices have already been used for this purpose [28]. Since improvement is affected by various mutable factors [29], it requires acquiring knowledge about the mutable situations on a continuous basis. CMMI [30], Experience Factory [31, 32], DevOps [5], and TQM and Lean [28] are examples of continuous improvement approaches. Table 3 provides a high-level view of how these prominent approaches cover the various phases of the KM process; as shown, fine-grained practices have not been prescribed by most of these approaches. We have obtained these results by comparing the techniques proposed by QI approaches to the goals pursued by the four phases of the KM process (Identification, Extraction, Sharing and Monitoring, of knowledge resources). The coverage-assessment results shown in Table 3 are explained below:

Table 3. Coverage of KM process by prominent QI approaches

QI approach	KM process			
	Identification	Extraction	Sharing	Monitoring
DevOps	⊡	■	◘	◘
Experience Factory	◘	◘	■	⊡
TQM	⊡	◘	⊡	⊡
CMMI	⊡	◘	⊡	■
Lean	⊡	◘	◘	⊡

□: Corresponding KM phase is not supported.

■: Fine-grained mechanisms are proposed to support the corresponding KM phase.

◘: Coarse-grained mechanisms are proposed to support the corresponding KM phase.

⊡: No mechanism is proposed to support the corresponding KM phase, but an appropriate Infrastructure/environment is provided to support the corresponding KM phase.

- DevOps: (1) *Identification*: The identification of developers (as knowledge resources) is not forced, even though continuous development is forced; thus, identification of knowledge resources is not forced, but is possible; (2) *Extraction*: Gaining fast feedback supports knowledge extraction; (3) *Sharing:* Practices for embedding development into operations (and vice versa) are prescribed, but fine-grained informal knowledge sharing mechanisms have not been proposed; and (4) *Monitoring*: Injecting quality gates is forced but fine-grained aspects of quality are not discussed.
- Experience Factory: (1) *Identification*: Characterizing projects and environments helps find the resources at a high level; (2) *Extraction*: Data analysis is conducted to find the practices, improvements, and problems. As a result, some types of knowledge can be extracted; (3) *Sharing*: An experience base is developed to share knowledge; and (4) *Monitoring*: Setting quantifiable goals prepares the environment for monitoring, although it is not forced.
- TQM: (1) *Identification*: Supporting leadership provides the environment for identifying the leaders that possess valuable knowledge resources; (2) *Extraction*: Facilitating self-improvement helps meet the prerequisites for extracting knowledge; (3) *Sharing*: Facilitating staff communication prepares the environment for sharing knowledge; and (4) *Monitoring*: Constant improvement of plans prepares the environment for monitoring.
- CMMI: (1) *Identification*: The prescriptions are focused on improvement, but some resources might be identified throughout the improvement steps; (2) *Extraction*: Decision analysis supports extracting hidden knowledge; (3) *Sharing*: Organizational training provides the environment for sharing knowledge; and (4)

Monitoring: Process and product quality assurance are emphasized and continuous monitoring is performed to improve the level of maturity.

- Lean: (1) *Identification*: Considering the people, product, and purpose dimensions provides the environment to find the knowledge resources; (2) *Extraction*: Analyzing root causes helps extract new knowledge; (3) *Sharing*: Collecting data about measures sets the stage for sharing knowledge; and (4) *Monitoring*: Controlled goal achievement is advised, and collecting data about measures is encouraged, but no specific techniques are provided for these purposes.

3 Research methodology

The main question that needed to be answered in this research was: How should a SME method build a SDM to flow the knowledge acquired throughout enactment of the methodology and software building processes? In turn, in order to build a method that supports the management of SME knowledge, three main questions should be answered: (1) Which weaknesses of existing SME methods should be targeted? (2) Which strengths of existing SME methods can be reused? and (3) Which requirements, in general, should be satisfied by SME methods? To answer these questions, our proposed SME method was iteratively built and evaluated through the following three steps (Fig. 2.), which respectively helped answer the three questions mentioned above:

1. *Alleviation* The first step was performed to alleviate the weaknesses of the eight SME methods reviewed in the previous section. For this purpose, a set of criteria were applied to these methods, and the following problems were revealed as their main shortcomings: deficiency in establishing an appropriate communication and collaboration environment, deficiency in prescribing comprehensive structures for sharing SME knowledge, and lack of support for operationalizing the engineered SDMs. Improved versions of these methods were then produced to address these shortcomings. Due to limitations in space, and also to observe the regulations of double-blind review, the complete set of criteria has been made available online as a supplementary file [8].
2. *Integration* In the second step, a SME framework (demonstrated in Fig. 3) was proposed by applying abstraction to the improved versions of the above-mentioned SME methods. As shown in Fig. 3, the proposed framework encompasses several umbrella activities (shown within the arrow symbol); these activities signify the tasks that should be performed in tandem with the following three iterative engineering phases:
   - *Preparation* With the aim of preparing the prerequisites needed for building the target SDM, three stages are prescribed: (1) making preliminary plans, preparing the resources required and scheduling the tasks, (2) analyzing the situational factors that affect

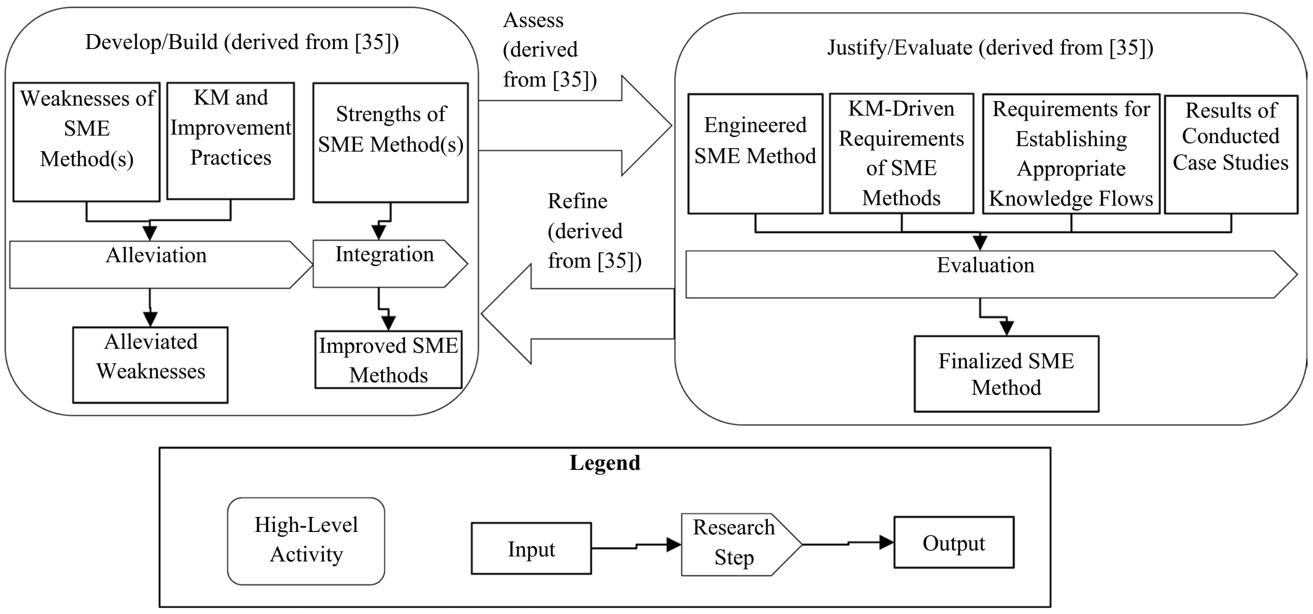
Develop/Build (derived from [35])
Weaknesses of SME Method(s)
KM and Improvement Practices
Strengths of SME Method(s)
Alleviation
Integration
Alleviated Weaknesses
Improved SME Methods
Assess (derived from [35])
Refine (derived from [35])
Justify/Evaluate (derived from [35])
Engineered SME Method
KM-Driven Requirements of SME Methods
Requirements for Establishing Appropriate Knowledge Flows
Results of Conducted Case Studies
Evaluation
Finalized SME Method
Legend
High-Level Activity
Input
Research Step
Output

Fig. 2 Research methodology (generally inspired by [33])

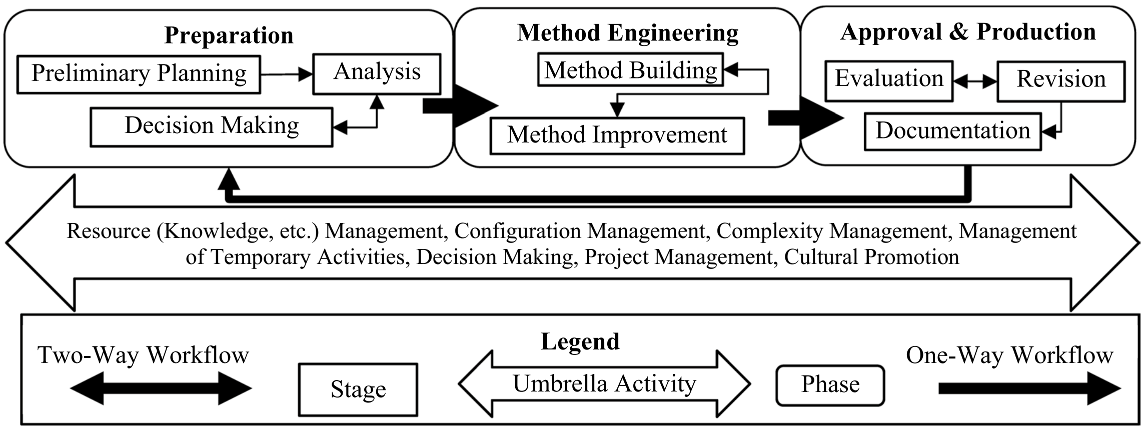

Fig. 3 Proposed KM-driven SME framework

the features of the target SDM [34], and (3) deciding on the construction strategy and the models that will be produced.

- *Method Engineering* By using the strategy that was selected in the previous phase, either a new SDM is built, or the existing one is improved.
- *Approval and Production* The engineered SDM is evaluated and put into production. It will then be continuously revised based on the feedback received from the users. The evaluation results and the lessons learnt are documented.

3. *Evaluation* The proposed method was iteratively improved by assessing its capability to satisfy the requirements pertaining to managing SME knowledge, and also to establish appropriate knowledge flows. In addition, four case studies were conducted to evaluate the proposed method in practice. The detail is provided in the following sections.

It should be noted that the high-level process of our research methodology (Fig. 2) has been inspired by the following guidelines, which have been proposed by Hevner et al. [33]: (1) “Design as an Artifact” which was applied by producing the SME method as an artifact, (2) “Problem Relevance” which was considered by using SME and knowledge sharing techniques to address the problems with available SME methods, (3) “Design Evaluation” which was addressed by evaluating the proposed method through conducting case studies and applying evaluation criteria, (4) “Research Contributions” which were described clearly by specifying the proposed method constituents, (5) “Research Rigor” which was addressed by considering the SME and KM foundations (which were encompassed in the KM-driven requirements of SME methods) and conducting case studies (by collecting data from experts), (6) “Design as a Search Process” which was conducted by finding the desired features of the SME method in the form of KM-driven requirements of SME methods, and (7) “Communication of Research” which was realized by presenting the proposed method through depicting its high-level phases and explaining the detail. We have used Hevner et al.’s guidelines due to the following opportunities that they provide: (1) evaluating various versions of the contribution (SME method) iteratively, (2) considering environmental factors (in terms of requirements for establishing knowledge flow), (3) checking the consistency with (KM and SME) foundations, (4) checking the applicability of the proposed contribution in different projects (companies), and (5) attention to the use of appropriate communication mechanisms to present the detail (fine-grained SME tasks).

4 Proposed SME method

Our proposed SME method consists of three parts: Process, Product, and People. These constituents are described in the following subsections. To use the proposed method, the roles prescribed in the "people" part should take the "process" steps to produce the "products".

4.1 Process part of proposed SME method

The proposed SME method has been engineered through instantiating the general KM-driven SME framework proposed in the previous section (Fig. 3), and then embedding appropriate KM and DevOps practices into it. As shown in Fig. 4, the process part of the proposed SME method incorporates two intertwined cycles, through which the target methodology is iteratively developed and operationalized. The correspondence between phases of the general SME

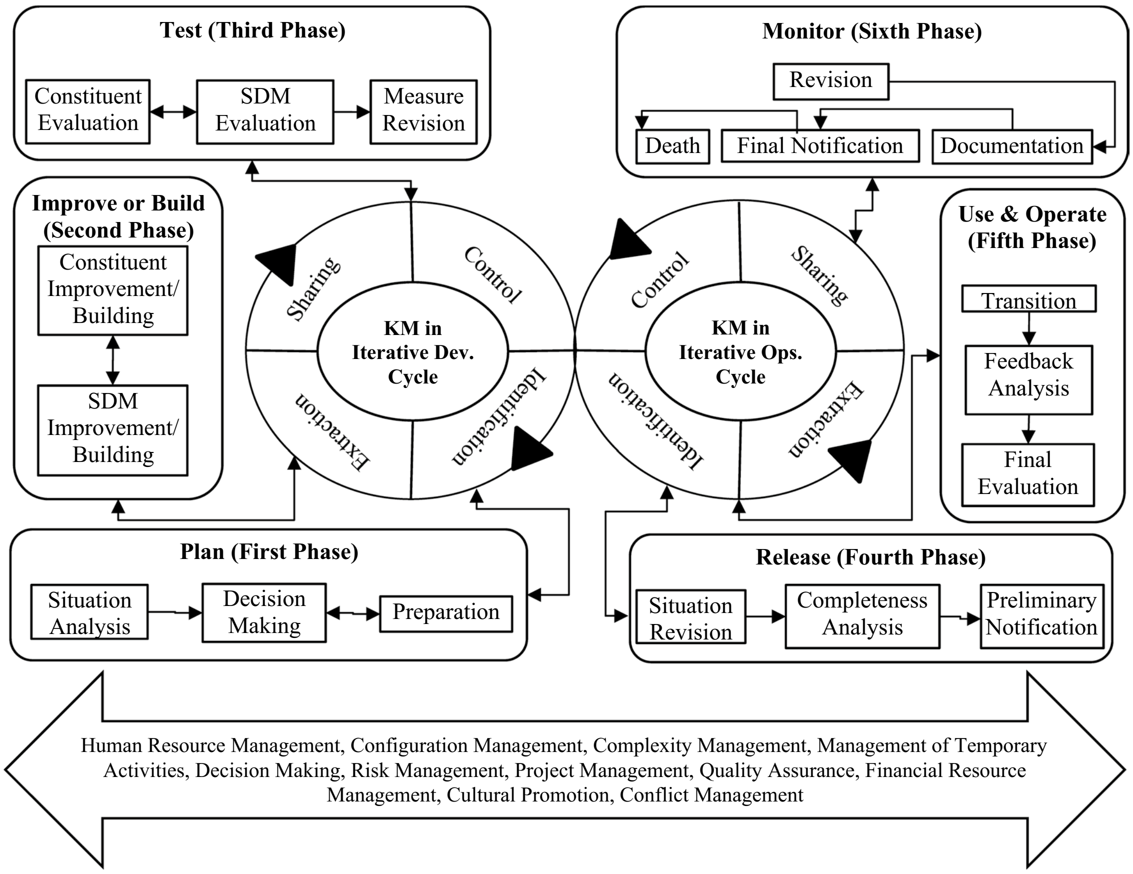
Test (Third Phase)
Constituent Evaluation
SDM Evaluation
Measure Revision
Monitor (Sixth Phase)
Revision
Death
Final Notification
Documentation
Improve or Build (Second Phase)
Constituent Improvement/ Building
SDM Improvement/ Building
KM in Iterative Dev. Cycle
Sharing
Control
Identification
Extraction
KM in Iterative Ops. Cycle
Control
Sharing
Extraction
Identification
Use & Operate (Fifth Phase)
Transition
Feedback Analysis
Final Evaluation
Plan (First Phase)
Situation Analysis
Decision Making
Preparation
Release (Fourth Phase)
Situation Revision
Completeness Analysis
Preliminary Notification
Human Resource Management, Configuration Management, Complexity Management, Management of Temporary Activities, Decision Making, Risk Management, Project Management, Quality Assurance, Financial Resource Management, Cultural Promotion, Conflict Management

Fig. 4 Proposed KM-driven and DevOps-based SME method

framework and stages of the proposed SME method is as follows: "Preparation" phase is refined into "Plan" stage; "Method Engineering" is realized by the "Improve or Build" stage; and "Approval and Production" phase is refined into "Test", "Release", "Use & Operate", and "Monitor" stages. In addition, KM practices are performed in parallel with Development and Operations cycles, and umbrella activities (shown on the arrow) are performed throughout the whole process. The following subsections describe these constituents.

4.1.1 Development (Dev.) Cycle

The target SDM is planned, built, and tested through the following stages (shown in Fig. 5):

1. Plan: Perform the following three stages:
   a. *Situation Analysis:* By using the characteristics provided in Table 4, analyze the characteristics of the situation for which the target SDM should be engineered; it should be noted that due to limitations in space, the complete version of the table is provided in the supplementary file [8]. To prepare the table, we have reused the characteristics introduced by Khatibian et al. for this purpose [35]; Khatibian et al. have defined these characteristics to categorize organizations at five levels of maturity as to their support for the KM process. We have reused these characteristics to specify the level of maturity in managing SME knowledge (rows of Table 4). Since SME knowledge encompasses three main categories of knowledge, the columns of Table 4 refer to the dimensions of SME knowledge. This helps answer the following questions from a KM point of view: (1) *What* are the factors that affect the SDP (software engineering knowledge)?; (2) *How* should these factors be addressed (method engineering knowledge)?; and (3) *Who* should perform the management necessary to address these factors (management knowledge)? To further elaborate, each of these dimensions incorporates the following knowledge contents:

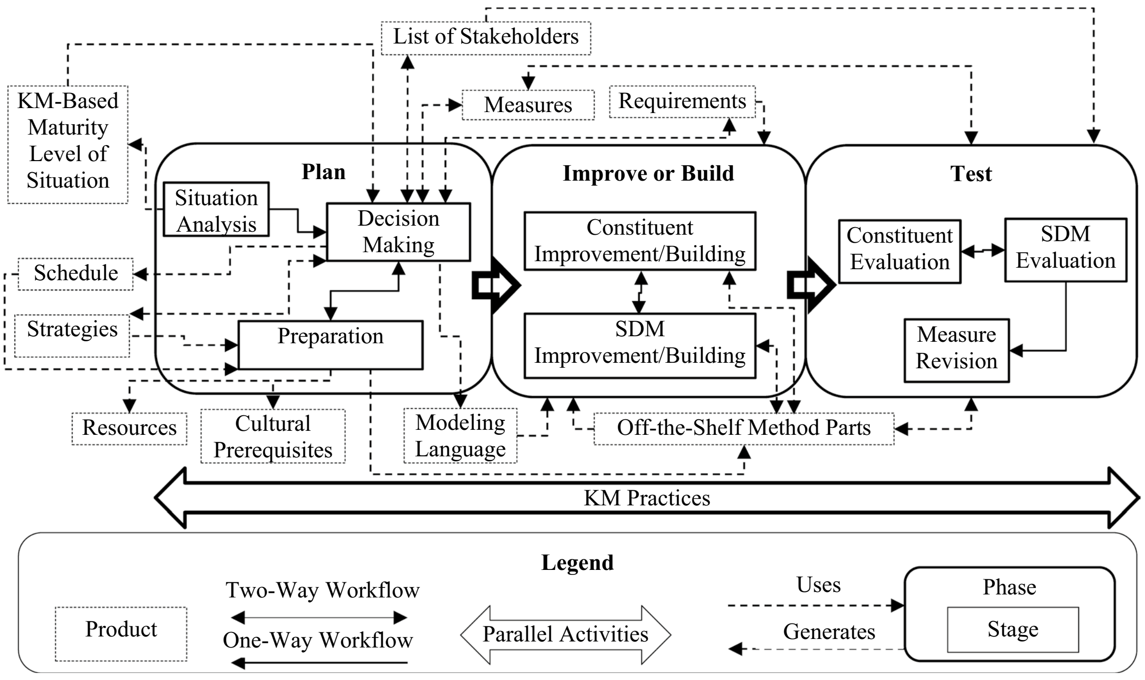

Fig. 5 Phases within the Dev. Cycle

What: Two categories of knowledge affect the software development process: (1) general (context-independent) situational factors within the following eight classes: "Business", "Application", "Management", "Requirements", "Technology", "Personnel", "Operation", and "Organization" [34], and (2) special (context-dependent) situational factors, which depend on the paradigm/context through which the software is developed. For example, various agent-oriented methodologies have been engineered and the SME process should guide creating/choosing the most appropriate one. These methodologies vary in four categories of criteria, namely "Process-Related", "Technique-Related", "Model-Related", and "Supportive-Feature" [36].
How: The way a SDM is engineered should describe the engineering process steps, and also the Modeling Language (ML). The ML specifies the rules for producing the artifacts of the process part.
Who: People who have responsibilities for method engineering should be skilled enough in the Software Engineering (SE) and Method Engineering (ME) areas of knowledge related to the situation at hand.

b) *Decision Making:* Specify the following items:

Inconsistencies: Check the consistency between in-use and to-be-used practices. The mutual relationships between in-use management approaches and KM practices should thus be investigated.
Stakeholders: Identify the stakeholders of the target SDM. Method engineers, who improve/build the SDM, and software developers who use the SDM are examples of stakeholders.
Requirements: Specify three categories of requirements by analyzing the features required for improving the level of maturity of (1) target SDM, (2) SME method, and (3) skills of human resources. Stakeholders should be involved in this step.
Strategies: Henderson-Sellers et al. have proposed various strategies for SME [2]. We propose a new KM-driven and assembly-based strategy. This is an evolved version of the assembly-based SME strategy, which takes advantage of the "Methods as Action Knowledge" viewpoint [2]. In this strategy, each SDM is imagined as a multi-dimensional knowledge block which is itself composed of smaller knowledge blocks, called method parts. As shown in Fig. 6, each method part incorporates three dimensions:

1. Content: encompasses the three dimensions of SME knowledge analyzed in Table 4. For example, a method part might force project managers (Who) to schedule a repetitive task for assessing (How) developers' knowledge about technology (What). Knowledge about technology is a general situational factor that affects the SDP [34].

Table 4 Dimensions and levels for situation analysis – A brief overview of main characteristics

	Dimension		
Maturity level	What	How	Who
Initial	No clear definition for knowledge contents in SDMs	No clear definition for knowledge contents in SME methods	No plan for managing SME process Incomplete understanding of SME concept
Managed	Identifying knowledge assets in SDMs Structural support for KM process (by SDMs) Specifying goals for SDMs	Identifying knowledge assets in SME methods Structural support for KM process (by SME methods) Specifying goals for SME methods	Specifying responsibilities for SE and SME Considering the priority of SE and SME goals Support for training, rewarding, and motivating software and method engineers Valuing software and method engineers' knowledge
Defined	Technical investigation of SE requirements Detailed description of SE process Embedding valuable SE experience into SDMs Defining KM sub-processes within SDMs	Technical investigation of SME requirements Detailed description of SME process Embedding valuable SME experience into SME methods Defining KM sub-processes within SME methods	Emphasize on managing human resources' knowledge Provision and revision of metrics for evaluating performance of SE and SME processes Defining KM responsibilities throughout SE and SME processes Plan for training software and method engineers
Quantitatively managed	Using quantitative measures for assessing performance of KM sub-processes in SE process Focus of SDMs on leadership and assessment	Using quantitative measures for assessing performance of KM sub-processes in SME process Focus of SME methods on leadership and assessment	Making improvement plans Monitoring and revising KM sub-processes Fostering an atmosphere of open communication Creating a collaboration culture
Optimizing	Continuous improvement of SE process Using modern SE technologies Sharing SE practices experienced Using a qualified method for measuring performance of SE process Embedding KM sub-processes in SE process appropriately	Continuous improvement of SME process Using modern SME technologies Sharing SME practices experienced Using a qualified method for measuring performance of SME process Embedding KM sub-processes in SME process appropriately	Choosing appropriate mechanisms throughout SE and SME processes Well-establishment of knowledge sharing and innovation culture Focus of KM plans on culture Benefiting from innovative and creative characteristics of method and software engineers Alignment of KM sub-processes with SE and SME goals Establishment of a learning culture Performing KM sub-processes routinely

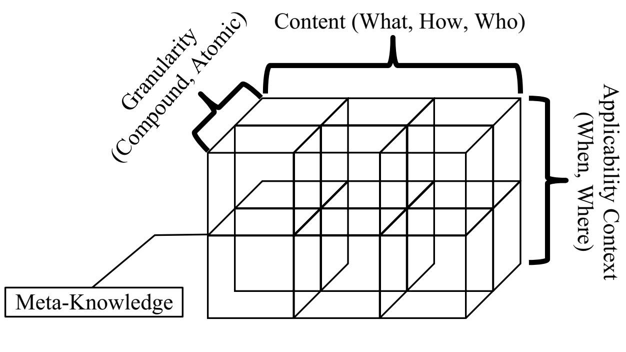
Content (What, How, Who)
Granularity (Compound, Atomic)
Applicability Context (When, Where)
Meta-Knowledge

Fig. 6 Knowledge dimensions (Cube) of method parts

To address this issue, the method part has specified a responsibility (continuous assessment of knowledge) for a role (project manager).

2. Applicability Context: specifies the time and the place (case) in which applying the method part is recommended. For example, assessing developers' knowledge about technology (the above-mentioned method part) should be performed in organizations/projects that deal with changing/new technologies.
3. Granularity: specifies if the method part is atomic or incorporates finer-grained parts. For example, assessing developers' knowledge (prescribed by the above-mentioned method part) might be performed directly by assigning tasks, or indirectly by checking the resumes. Thus, the method part for assessing knowledge is a compound one.

Besides, meta-knowledge is used for providing complementary information, if required. In this step, method engineers should decide on the knowledge dimensions which should be specified/updated in the current Dev. and Ops. cycles. Deciding about the strategy, and thus the knowledge dimensions, facilitates the sharing of SME knowledge by standardizing the structure of the knowledge shared. Otherwise, SME techniques would have to be pooled in various forms, which would make them difficult to grasp; a deficiency that afflicts all existing SME methods.

It should be noted that the proposed knowledge cube (Fig. 6) has been inspired by the efforts previously made in the area of both SME and KM. The problem with available SME methods is that they do not encompass all the dimensions in detail. Most available methods specify the content, but do not specify the *corresponding context* and *levels of granularity*. For example, the SME methods that are proposed based on the assembly-based approach specify the granularity of method contents, but they do not necessarily encompass all the dimensions of the content (what, how, who); also, it is not mandatory to specify both the time and the place (case) of using the content. Putting all three dimensions together, as well as specifying the detail in a unified structure (What, How, Who, When, Where, Granularity Level), are the novelties of this approach.

Measures: Define measures to assess the quality and completeness of current Dev. and Ops. cycles. These measures are inspired by requirements, strategies, and features of the current level of maturity specified in the previous steps. Method engineers should be asked to commit to this definition. This step has been inspired by the Definition of Done (DoD) technique [37]. It should be noted that the duration of the development and operationalization cycles is dependent on the measures decided upon; thus, this item is not fixed beforehand.
ML: Select an appropriate ML to present the knowledge dimensions of artifacts. Some examples of modeling approaches are: (1) Ontology-based technique [2, 38], and (2) process meta-models, such as SPEM 2.0 [39].
Schedule: Prepare a timeline to schedule current Dev. and Ops. cycles. This timeline should be consistent with the specified requirements and measures.

c) *Preparation:* Establish an appropriate collaboration and learning environment, and ask method engineers to discuss the rationale behind the tasks and changes prescribed. Ask individuals to choose the tasks themselves and prepare the following items: (1) off-the-shelf method parts which might be reused; (2) prerequisites for alleviating cultural barriers; and (3) other resources required. Team separation, lack of a common language between different roles, and fear of sharing knowledge [5] are examples of cultural barriers.

2. *Improve/Build* Pursue the following stages to build/improve the target SDM:

    a) *Constituent Improvement/Building:* Build a new method part, or update an existing one. Previously engineered method parts (off-the-shelf method parts) might be reused in this step. It should be noted that the rationale behind selecting/building method parts should be preserved through updating knowledge dimensions of selected/built method parts. Also, method parts should be traceable to requirements.
    b) *SDM Improvement/Building:* Integrate chosen method parts to build/improve the target SDM. Developers should be involved in this step to provide feedback about the method they should use. All versions of the under-construction SDMs and their corresponding knowledge cubes should also be shared in order to be reused in the future.

3. Test: Use previously specified measures to evaluate: a) the method parts, and b) the current version of the engineered SDM. Finally, use stakeholders' opinions to revise the measures.

4.1.2 Operations (Ops.) Cycle

The following three phases are performed to operationalize the developed SDM (shown in Fig. 7):

1) *Release* Release the current version of the SDM through the following stages:

    a) *Situation Revision* Investigate changes in characteristics of the situation for which the SDM is being engineered, and specify changes in the level of maturity.
    b) *Completeness Analysis* Assess the satisfaction of requirements targeted at current Dev. and Ops., cycles, and revise the requirements list, if required.
    c) *Preliminary Notification* Release the current version of the engineered SDM, and report on these items: reasons for postponing delayed tasks, number of non-satisfied requirements, and current level of maturity.

2) *Use and Operate* Put the engineered SDM into practice through the following stages:

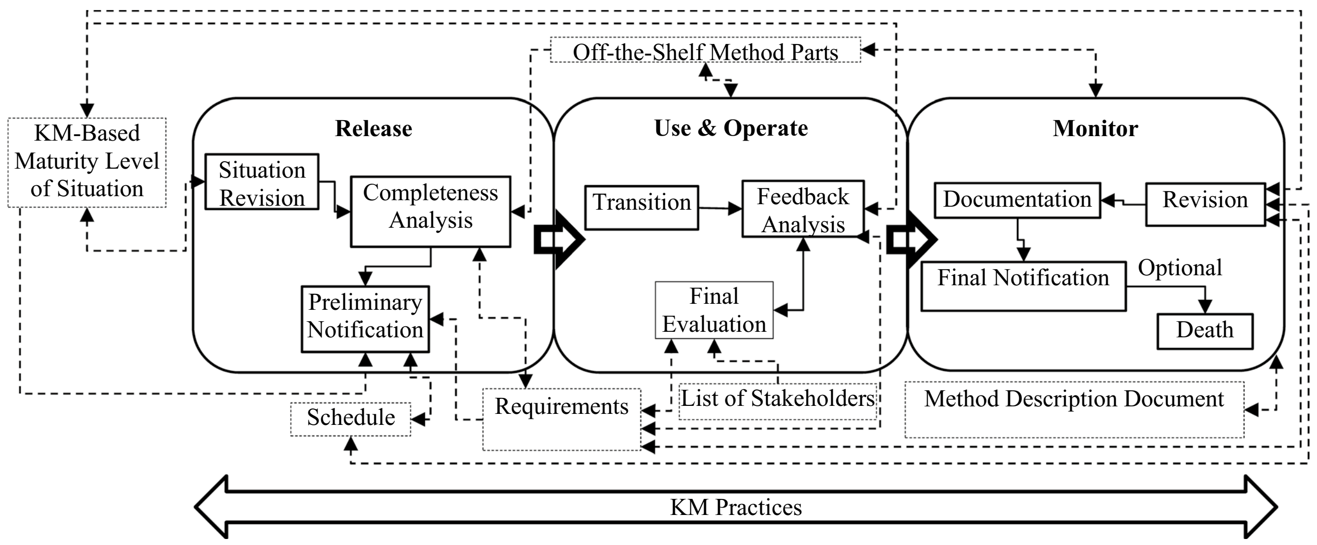

Fig. 7 Phases within the Ops. Cycle

a) *Transition* Ask customers (developers) to use the current version of the engineered SDM. If they have to switch between new and old versions of the SDM, it is recommended to use the "Blue-Green Deployment" technique [5]; in this way, the new version will be used by some of the developers in the blue environment, while other developers use the old version. The "Dark Launching" technique [5] might also be used to receive feedback from handpicked individuals. These feedbacks should be saved and shared.

b) *Feedback Analysis* Analyze feedbacks received in the previous phase, and specify new requirements. Then, report on changes in the level of maturity of the SME method, if applicable.

c) *Final Evaluation* Ask stakeholders to use the engineered SDM, and evaluate their satisfaction with the current version. Define new requirements, if necessary.

3) *Monitor* During the time that the engineered SDM is being used, follow these stages to prevent it from deterioration:

a) *Revision* Reassess the situational factors and find new requirements to improve the engineered SDM through re-performing Dev. and Ops. cycles. Then, revise the in-use SME method, and ask the method engineers to change their responsibilities to blur roles [5]. Finally, plan for using appropriate KM practices to prevent the occurrence of the following anti-patterns:

"Hero cult" [5]: All the individuals should share their knowledge, and heroes should not be created.

"Emphasis on titles" [5]: Individual's skills should be evaluated based on the knowledge they share, rather than their roles in the SME process.

"Shadow responsibilities" [5]: All responsibilities should be monitored to be performed accurately (just as defined).

"Favor a plan over planning" [5]: Both the plan and the quality should be important for individuals. Schedules might be updated to ensure quality.

b) *Documentation* Document the results of the revisions conducted. Results of revising the SME method should be recorded in the SME method description document. Results of revising the SDM should also be saved as a kind of knowledge contained (knowledge-content) within the method parts.

c) *Final Notification* Report on the successes and failures documented in the previous stage. This helps motivate method engineers through appreciating their efforts for successes and also informing them about weaknesses.

d) *Death* If the in-use SME method does not satisfy the new requirements and it cannot be improved, stop using it. Record the lessons learnt (in the SME description document and the method parts).

4.1.3 Umbrella activities

The project manager and knowledge manager, described in the next section, are responsible for the umbrella activities. The supported umbrella activities are as follows:

- *Human Resource Management* Engaging method engineers and developers in SME activities along with pronouncing knowledge sharing awards supports training and assessing human resources.
- *Configuration Management* Versioning is applied to shared knowledge contents. In addition, changes are managed.
- *Complexity Management* This activity is supported by dividing complex methods into simple knowledge-contained method parts and using a gradual engineering process.
- *Management of Temporary Activities* Activities' lifetime is managed through sharing experience about starting and stopping challenges in running the activities and also by receiving feedback about the applicability of activities.
- *Decision Making* Decisions are made with more confidence by facilitating access to the right knowledge at the right time [40].
- *Risk Management* The iterative-incremental engineering process supports finding and alleviating the risks gradually.
- *Project Management* Iterative-incremental planning and monitoring of SME activities supports project management.
- *Quality Assurance* Quality assurance is supported by the activities within the "Test" and "Monitor" phases along with defining measures during the "Plan" phase.
- *Financial Resource Management* Financial resources, required for the SME process, are prepared and revised in the "Plan" phase.
- *Cultural Promotion* Cultural concerns are gradually addressed in the "Plan" phase.
- *Conflict Management* Conflicts between the engineered and in-use versions of the SDM are managed through defining joint development-operationalization responsibilities for some of the roles. Moreover, managing changes supports managing conflicts.

4.1.4 KM Practices

We have previously found the requirements for managing SME knowledge, and have presented them through a set of criteria which have been provided in the supplementary file [8]. Knowledge engineers can use these criteria to assess the accurate flow of SME knowledge. Appropriate KM practices should then be applied to satisfy the criteria required. For each type of elicited criteria, Table 5 provides examples of the criteria and the corresponding KM practices that satisfy the criteria.

4.2 People part of proposed SME method

The following roles take the responsibilities for performing the proposed SME process; Dev. and Ops. responsibilities are not separated, and individuals should choose their responsibilities themselves [5]:

- *Method Engineer* Activities that require ME skills, such as situation analysis, are performed by this role. Thus, individuals who take this responsibility participate in all the prescribed phases. They should also manage configurations of method parts.

Table 5 Examples of criteria and KM practices

Criterion ID (available in the supplementary file [8])	Criterion (Requirements for Managing SME Knowledge)	Solution (KM Practice)
SME-P 34	Prescribing appropriate mechanisms to use the experience available in the field of software engineering	Embed the capability for automatic KM in collaboration tools. For example, embed the capability to "identify an expert capable of answering a query" in in-use tools [41]
SDM-P 47	Prescribing appropriate mechanisms that support learning from software development activities	Use the Systemic Lessons Learned Knowledge (Syllk) model to facilitate learning from projects [42]
SDM-ML 8	Prescribing appropriate mechanisms to use appropriate models that support KM-driven extraction of the requirements of the target SDM	Use dynamic methods to extract and classify requirements-related knowledge [43]
SME-ML 21	Support for managing reusability knowledge	Provide the capability to choose a set of appropriate reusability practices such as the DevOps practices [44]
KIP 108	Support for collaboration-dependent knowledge-intensive SME processes	Use collaborative KM practices such as the practices provided in [45]

- *Developer* The developers who use the SDM should participate in the following stages: Final Evaluation, Transition, Decision Making, and Preparation. This helps define the requirements and evaluate the engineered SDM more accurately.
- *Knowledge Engineer* This role takes the responsibility for choosing appropriate KM practices. In addition, monitoring the process for establishing appropriate knowledge flows is fulfilled by this role. Establishing shared values and goals is also assigned to this role.
- *Project Manager* A method engineer who is familiar with KM practices should play this role and direct other individuals. This role is also responsible for scheduling and configuring the SME process.

4.3 Product part of proposed SME method

The following outputs are produced/updated throughout the proposed SME process:

- *KM-Based Maturity Level of the Situation* This includes the characteristics of the situation, analyzed in the first phase.
- *Schedule* Tasks, deadlines, and responsibilities are recorded in this product.
- *Strategies* As previously explained, our method uses the assembly-based strategy, but it does not force preserving all the knowledge dimensions of the method parts assembled. Valuable knowledge dimensions of the method parts are discussed in this product.
- *Resources* The results of analyzing the resources are reported in this product. This includes tools, miscellaneous knowledge contents (which are not preserved within the method parts), and also financial and human resources.
- *Cultural Prerequisites* All the prerequisites required for supporting the cultural dimensions of KM-driven SMEs are explained in this product.
- *ML* This product contains syntax and semantic rules for producing the method parts.
- *Off-the-Shelf Method Parts* This product refers to reusable (previously engineered) method parts.
- *List of Stakeholders* This list provides information about the individuals who are affected by, or affect, the target SDM.
- *Measures* This product provides information about the criteria used for assessing the level of maturity in support of the KM process.
- *Requirements* Requirements specify the features required to be supported by the SME method or the target SDM.
- *Method Description Document* This document describes the SME method in detail.

5 Evaluation of proposed method

Our proposed method was assessed through criteria-based evaluation and a case study, explained in the following subsections.

5.1 Criteria-based evaluation

We have previously elicited a set of 381 criteria to assess two issues: (1) ability in managing SME knowledge, and (2) support for engineering SDMs which are themselves capable of managing knowledge throughout the SDP; as mentioned before, due to limitations in space and also because of the regulations of double-blind review, these criteria has been made available online as a supplementary file [8]. We have used these criteria to compare our proposed SME method with existing ones. We have found out that our proposed method is superior to existing SME methods in the following aspects:

1. *Supporting adoption of the engineered SDM:* Adopting SDMs, even light ones, might be difficult [46]. The proposed "Preparation" stage addresses this issue by defining prerequisites for sharing adoption knowledge. For example, problems with forming self-organized teams are alleviated by teaching communication and collaboration skills and also by monitoring cultural obstacles.
2. *Support for engineering comprehensive SDMs:* As previously mentioned, previous methods have focused on the process aspect of SME knowledge. The proposed method provides the opportunity to capture and share various dimensions of knowledge (Fig. 6). This helps preserve the rationale behind selecting and integrating method parts and thus prevent engineering stovepipe SDMs.
3. *Preventing early death of the engineered SDM:* KM practices are performed all through the DevOps cycles. This helps monitor and update the engineered SDM continuously, and thus improve it when required.
4. *Supporting definition of visions:* Analyzing the level of maturity of situations helps define visions for in-use SDMs.

Besides the above-mentioned strengths, our method suffers from the following weaknesses:

1. *Lack of a comprehensive ML:* We have not proposed a language for modeling the products of the SME process.
2. *Lack of a comprehensive SME tool:* Method engineers should themselves decide about the tools that they will use for SME. EPFC is an example of such tools [47].

3. *Complexity:* In comparison with previous methods, our method provides more detailed instructions. In addition, both DevOps and KM encompass a large number of practices. Even though we have only used the practices that help address the weaknesses of available SME methods, the proposed method is still larger and more complex than its predecessors. To address this issue, constituents have been classified into different levels of abstraction, and we have used a top-down approach to describe the proposed method.
4. *Need for standard measures:* Method engineers should specify appropriate measures for evaluating both SME methods and SDMs. These measures are affected by the method engineer's goals and knowledge, and we have not provided a standard set of measures for this purpose yet.

5.2 Case study-based evaluation

With the aim of validating the proposed SME method and finding its strengths and weaknesses (as perceived by method engineers and developers), four case studies have been conducted based on the guidelines provided in [48–52]. We have also used the case studies reported in [53–55] as templates for reporting our case studies. Reports of the four case studies are provided in the following subsections.

5.2.1 Case study design and planning

With the aim of evaluating the proposed method in practice, two main questions were targeted: (1) What are the strengths and weaknesses of the proposed SME method? and (2) What are the aspects in which the SDMs engineered by applying the proposed SME method are superior/inferior to the SDMs previously in use? As mentioned, the responses should encompass the strengths and weaknesses from both method engineers' and developers' points of view.

To find the answers, the cases were selected based on the following criteria [51]:

1. *"Maximum Variation"* [51]: Companies with SME processes at different levels of maturity were selected; the maturity level was determined by using the information provided in Table 4. A brief review of the features of these companies is provided in Table 6; the names are intentionally kept anonymous. Also, Table 7 shows a general overview of the main strengths and weaknesses of the companies' in-use processes. A customized version of the Scrum methodology was used in all the chosen companies, but the level of capability for KM was not the same across these companies. It should be noted that Table 6 does not show the size of each company, but the size of the unit in which the case study was conducted.

Table 6 Features of the companies chosen as case study venues

Name (Unit)	Business Area	In-Use Methodology	Age	Size	Level of Maturity		
					"What"	"Who"	"How"
A (One Unit, Two Sections)	IT	Scrum	25	11	3	1	1
B (Four Units)	IT	Scrum	17	20	3	2	3
C (One Unit)	IT	Scrum	7	15	2	1	2
D (One Unit)-Startup	IT	Scrum	3	3	3	2	2

Table 7 Main strengths and weaknesses of in-use processes in the companies chosen as case study venues

Company	strengths	Weaknesses
A	Developers' tendency to improve in-use processes	Inability to create a friendly workplace
	Being informed about a large number of available knowledge resources	Inefficiency of motivational mechanisms
B	Ability to create a friendly workplace (in one unit)	Inefficiency of training mechanisms
	Speciation of measures for assessing products	Dissatisfaction with opportunities for career progression
C	Avoidance of structural limitations which prevent knowledge sharing	Being dependent on specific developers
	Developers' tendency to participate in workshops	Suffering from knowledge hoarding culture
D	Being familiar with appropriate mechanisms for sharing knowledge with customers	Inability to specify criteria for evaluating processes
	Ability to create a friendly workplace	Inability to update plans and make accurate estimations

2. *Availability of Knowledge* In some of the companies, acquiring information from resources was restricted; hence, they were excluded from the case studies.
3. *Perception of Necessity* We chose those companies in which managers felt the need for using KM-driven method engineering approaches. This has facilitated the knowledge sharing process.

5.2.2 *Data collection*

Data was collected in two rounds; the first round was for investigating the in-use method, and the second was for analyzing improvements and shortcomings. Semi-structured interviews and questionnaires were designed for this purpose. The main questions of the questionnaires and interviews are provided in Table 8 and Table 9. These questions were asked in all the companies. Unique questions were added for each company based on the specific problems encountered in that company. For example, when

Table 8 Designed questionnaire and corresponding target respondents

First Round of Questions					
Question (Target Respondent(s))	Strongly Agree	Agree	Neu-tral	Disa-gree	Strongly Disagree
I prefer to work alone rather than participate in team activities (D)					
I'm not worried about explaining the mistakes that I have made (D)					
Some skills are monopolized by specific persons (B)					
I can easily communicate with the managers (D)					
I am satisfied with the methods used for monitoring the processes (M)					
Changes required to be made to software development processes are not assessed (M)					
I am interested in team activities (D)					
Responsibilities are clearly defined (B)					
Standards for producing products are specified (B)					
Second round of questions					
I am now interested in sharing my experience (B)					
Certain documents that help avoid rework are now produced (M)					
I am now satisfied with the training techniques (B)					
I am now consulted before updating the workflows (D)					
The results of this case study helped analyze the strengths and weaknesses of in-use SDM (M)					
The results of our study helped you analyze the strengths and weaknesses of the in-use SME method (M)					
The Operations cycle helps engineer more operationalized SDMs (M). Please explain					
Iterative-incremental application of changes helps improve applicability of the revised version of the Scrum methodology (M)					
Performance of team-work efforts is improved by holding workshops about communication and collaboration skills (B)					
Notifying about the achievements and failures helps clarify the criteria for a job promotion (B)					
The work breakdown structure used for describing the SME method improved the practicability of the proposed method (M)					
Accurate specification of the products of the SME method helped preserve the project managers' and developers' knowledge (M)					
Planning software development processes is now performed more accurately (B). Please explain					
The iterations for building/revising the software development processes are now conducted more effectively (B). Please explain					
Processes are now evaluated more accurately (B). Please explain					
The iterative-incremental process for releasing the changes has improved the efficiency of changes (B). Please explain					

M Method Engineers;*D* Developers; *B* Both Developers and Method Engineers

Table 9 Designed interview questions and corresponding target respondents

First Round of Questions (Target Respondent(s))
Which SDMs have been used in this company (M)?
What are the strengths and weaknesses of the in-use processes (B)?
Which mechanisms are used to find the skills required (B)? How are the employees trained (B)?
Which techniques are used for motivating the developers (B)?
Which roles are involved in engineering and running the development processes (M)?
Which mechanisms are used for monitoring the SDMs (M)?
Which tasks are performed to develop a software (B)? How are these tasks customized (B)?
Which tools and documents are used to share the products throughout the software development process (B)?
Second Round Of Questions (target respondent (s))
Could you find and alleviate the problems with collaborations (M)?
What are the weaknesses and strengths of the proposed SME method, in your opinion (M)?
What are the weaknesses and strengths of the improved SDM, in your opinion (M)?
What is your opinion about the applicability of the improvements proposed to be applied to in-use Scrum methodology (B)?
What are the strengths and weaknesses of this case study in terms of attractiveness, timing, etc. (B)?
Do the motivational mechanisms, proposed by this method, helped increase employee performance (M)?
Are the roles, prescribed by the SME method, comprehensive enough to encompass all responsibilities (M)?
How much did it take for you to understand the prescriptions by the proposed SME method (B)?

M Method Engineers; *D* Developers; *B* Both Developers and Method Engineers

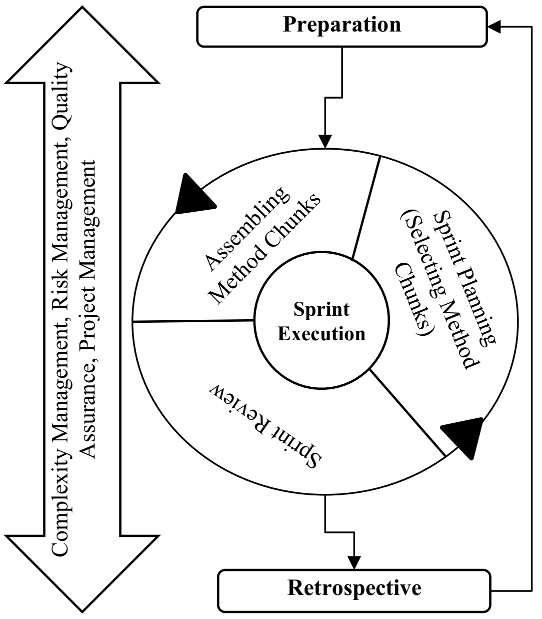
Preparation
Complexity Management, Risk Management, Quality Assurance, Project Management
Assembling Method Chunks
Sprint Planning (Selecting Method Chunks)
Sprint Execution
Sprint Review
Retrospective

Fig. 8 In-use SDM (a customized version of the Scrum methodology)

one of the managers reported problems with teamwork, we added more questions to find the causes for this issue.

5.2.3 Data analysis

The following subsections provide the results of qualitative and quantitative analysis of the data collected.

a) *Qualitative data analysis*

By analyzing the data collected, the in-use versions of SDMs were identified, and improved versions were then engineered. Figure 8 shows a general high-level schema of the in-use Scrum methodology [56] in all the companies. It should be noted that the fine-grained tasks were different among the companies, but the general process was the same. Figure 9 shows the new customized version of the Scrum methodology that was engineered by using our proposed SME method. A clearly visible improvement is that sprints have been revised so that KM practices are performed in parallel with development activities. Since developers should be motivated for using KM practices, the left-side sprint should first be performed by the time individuals start using KM practices unconsciously. In addition, more comprehensive umbrella activities, shown within the arrow symbol, have been prescribed to be performed in parallel with other activities. For example, appropriate cultural techniques have been prescribed for preventing developers from hoarding their knowledge. The second round of data was thus collected to assess the SME method that was used for engineering this methodology, and also to assess the customized methodology itself.

By analyzing the data collected, in addition to the features of the in-use SDMs, the following practical implications were revealed for the proposed approach:

- *Strengths (Positive Implications)* (1) improving in-use processes by emphasizing on using developers' experi-

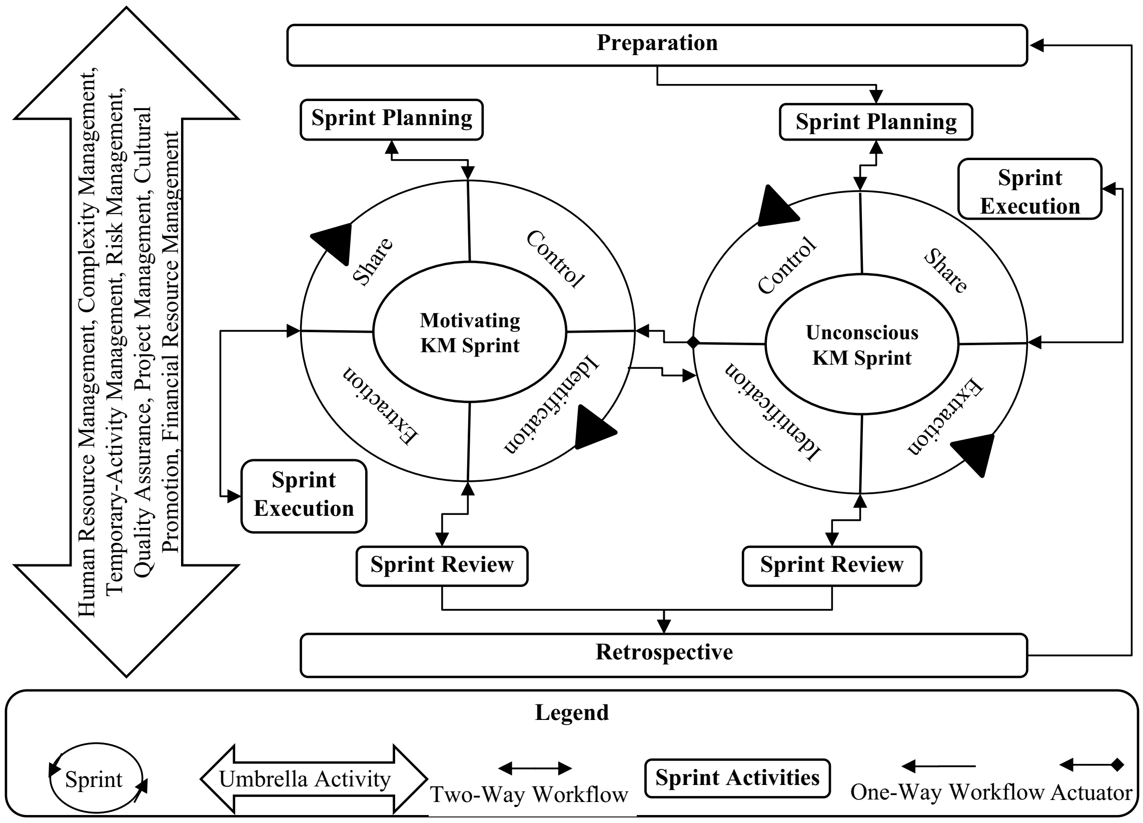

Fig. 9 Improved SDM (a tailored and improved version of the Scrum methodology)

ence in addition to method engineers' experience; (2) combining theoretical foundations with practical solutions by supporting experience-based method engineering rather than emphasizing theoretical-based method engineering; (3) supporting routinization by providing prescriptions for continuous improvement; (4) avoiding resistance to change by supporting continuous transition of engineered SDM instead of changing the in-use SDPs in a one-time fashion; (5) illustrating the as-is situation by providing the capability to provide quantified and detailed results for analyzing situations; (6) supporting training by providing instructions for analyzing the skills required for method engineering; (7) providing a unified comprehensive structure for describing various parts of SME methods through proposing a knowledge cube structure; and (8) facilitating maintenance by continuous revision of measures to maintain the usability of method parts.

- *Weaknesses (Negative Implications)* (1) dependency on certain experts due to the need for an expert knowledge engineer to facilitate the KM process; (2) late returns due to the time needed for applying the KM practices; (3) dependency to method engineers' capability to describe SDMs because of inability to prescribe a standard language for describing the engineered SDMs; and (4) need for a tool to search for available method parts because of their sheer multitude.

b) *Quantitative Data Analysis*

Developers were users of the engineered SDMs, but their expertise was not at the same level as that of method engineers. Thus, they could not conduct the evaluation directly. As a result, two types of questions were designed: (1) general questions, which indirectly assessed whether the subjects (developers and method engineers) agreed or strongly agreed with the solutions, and (2) specific questions, which were specifically designed for method engineers who had a deeper understanding of the prescribed solutions and could therefore directly assess them.

We have used Likert-type scales for quantifying the responses to the questions (Strongly Agree (2), Agree (1), Neutral (0), Disagree (-1), Strongly Disagree (-2)) [57]. The following methods have then been used for analyzing the data collected.

1. *Mean* We used Formula 1 to calculate the average of the responses. As shown in Table 10, the result of calculating the average of quantitative responses was posi-

Table 10 Results of statistical evaluation

Measure	Company				
		A	B	C	D
Mean	Developers' special questions	0.8	0.2	0.1	0.3
	General questions	0.6	0.5	0.4	0.7
	Experts' special questions	0.82	1	0.4	0.82
Precision		0.72	0.55	0.89	0.78
Internal consistency (Cronbach's α)		0.96	0.99	0.95	0.96

tive, which means that most solutions were confirmed by developers and experts. The experts' special questions helped compare our proposed method with previous SME methods. Since the mean for answers to these questions is positive, it can be claimed that most experts have confirmed that our method is superior to previous SME methods in both managing SME knowledge and engineering KM-driven SDMs.

5.3 Formula 1. Mean value of responses

$$\frac{\sum_{Qi=1}^{Qi=NumberofQuestions} \frac{\sum_{Ri=1}^{Ri=\text{Number }ofRespondents} ResponseinLikert' sScale(\text{Qi})}{2*NumberofRespondents}}{NumberofQuestions}$$

2. **Precision:** KM practices are context-dependent [58]. In some cases, the developers' perception is more precise than the method engineers', as the developers are the ones who actually apply the practices proposed by our method. Thus, developers can have a more precise context-dependent judgment in such cases. In order to assess the precision of the developers' agreement with the experts, the true positive rate of responses was calculated (Formula 2). To this aim, we assumed that experts' (method engineers') responses indicated the true (right) solutions, and then compared developers' and experts' responses by calculating the true-positive rate. As shown in Table 10, in most cases, developers confirmed the experts' opinions.

5.4 Formula 2. Precision of proposed practices

$$\frac{\sum_{Si=1}^{Si=Numberof\ \mathrm{Pr}actices} C(Si)}{Numberof\ \mathrm{Pr}actices},\ C(Si) = \begin{cases} 1, Consistencybetweendevelopers\prime and \exp erts\prime responses \\ 0, Inconsistencybetweendevelopers\prime and \exp erts\prime responses \end{cases}$$

3. **Internal Consistency:** The questions we have designed should measure the same issue, which is the efficiency of the prescriptions by the proposed SME method. To assess this issue, we have checked the internal consistency of the responses. This means that a respondent should not provide inconsistent responses which affect the reliability of the results of the assessment conducted. For example, if an employee prefers working alone, he/she cannot be interested in teamwork at the same time. We have thus calculated the Cronbach's α to analyze this issue (Formula 3). As shown in Table 10, in all the companies, this measure was greater than 0.9.

5.5 Formula 3. Cronbach's Alpha

$$\frac{N\complement}{v+(N-1)\complement}, N = Numberofrelatedquestions,\ \complement = Average\mathrm{int}erquestion\ \mathrm{cov}ariance, v = Average\mathrm{int}erquestion\mathrm{v}ariance$$

5.5.1 Reporting

Internal reporting of the results has been done through holding meetings by the managers. We had three main meetings for this purpose; sub-sessions were also held for guiding the employees. In the first meeting, the as-is situation of the company was described by explaining the results of analyzing the responses to the questions of the first rounds of interviews and questionnaires. The second meeting was aimed at informing the managers about the process of applying the changes and also asking for their help to alleviate the problems encountered. Finally, we reported the improvements in the third meeting, and also provided suggestions for future follow-ups.

We cannot disclose the artifacts of the case studies because of legal reasons. As mentioned above, we have masked the company's names and their private information for the same reasons.

5.5.2 Lessons learned

We have learned the following lessons through the case studies conducted:

- Plans should be flexible because changing the routines is difficult and requires managing the resistance to change. We found that some employees were worried about the results of our study; we therefore decided to have coffee meetings with them to explain the benefits of our study for their work.
- In some cases, analyzing the process documents and models has not provided concrete information, as some employees update these products at the end of each year. Thus, we had to collect the new information through interviews. In other words, we learned to assess the reliability of explicit knowledge resources before using them.
- Effort estimation should be conducted by collaborating with project managers. Since the managers have previous experience with controlling company projects, they can provide more accurate estimations. After analyzing the as-is situation, we asked the managers to check the estimations we had provided.
- Choosing appropriate data analysis methods requires considering the features of the data that will be available. We had to omit the unreliable data (which included inconsistent responses), and thus could not use Structural Equation Modeling (SEM) techniques that require a large amount of data.

5.5.3 Threats to validity

The four aspects of the validity of our case study have been assessed as follows:

- *Construct Validity* Various interpretations of the questions were investigated in the pilot study (by an expert), and the questions were then revised. Moreover, the questions were reviewed by the second author. Furthermore, we assumed that developers had the same interpretation of the questions, regardless of their roles in the SDP. To make sure about this assumption, the responses were randomly divided into groups, and single factor analysis was applied to assess common method bias [59]. We used the ANOVA data analysis method available in Excel for this purpose. Table 11 shows the results of the analysis. Since the values calculated for F are less than F critical, we can assume that the responses provided by different roles have the same variance; this indicates that the means for different groups' responses are equal.
- *Internal Validity* Individuals who filled the questionnaire were asked to specify the factors that had not been mentioned in the questionnaire, but might be useful. This helped find the factors that had been neglected in investigating the causal relations. Furthermore, to check the mean difference between responses, the t-test has been conducted. Since a few of the developers did not have adequate time to fill the questionnaire completely, we hypothesized that losing these responses would result in non-response bias. To assess this hypothesis, we classified the respondents into two groups, and then used the t-test method to assess non-response bias [60]. The results are provided in Table 11. We noticed that the following condition had not been met: "t Stat < -t Critical two-tail or t Stat > t Critical two-tail"; this means that the responses do not fall into the rejection region. Hence, our assumption was not true and losing some of the responses did not affect the overall results; thus, non-response bias was insignificant.
- *External Validity* Threats to this kind of validity were minimized because we tried to choose various cases; this helped generalize the results. Moreover, we used the triangulation technique to ensure the coverage of various kinds of data; this was achieved by asking a variety of individuals to fill the questionnaires. In addition, quantitative and qualitative analysis methods were both used, and both method engineers and developers were engaged in the evaluation process.
- *Reliability* During the pilot study, method engineers were asked to comment on the ambiguous aspects of the questions. Also, the collected data were analyzed by using well-known methods, and the analysis results were reviewed by the second author. We have thus strived to enhance the reliability and repeatability of the results.

Table 11 Single Factor and *t*-Test Analyses

Company	Single Factor Analysis			*t* Test Analysis		
	F	F Critical	*P* Value	$P(T <= t)$ two-tail	*t* Critical two-tail	*t* Stat
A	0	4.13	0.90	0.84	2.11	−0.28
B	0.6	2.31	0.91	0.94	1.95	0.51
C	0.8	1.65	1	0.81	1.78	−0.32
D	0.3	1.90	0.95	0.93	1.10	−0.1

6 Conclusions and future work

The KM-driven and DevOps-based SME method that we have proposed in this paper is different from previous SME methods in the following two main aspects: (1) analyzing and improving support for KM process, and (2) gradual operationalization of the engineered SDM.

The proposed method was evaluated both theoretically (based on evaluation criteria) and empirically (through case studies). The results indicate that the proposed method is strong in the following features: (1) involving people who might seek or have valuable knowledge about the project situation; (2) engineering SDMs with three main interwoven elements, namely people, product, and process; (3) improving in-use SDPs; (4) scrutinizing SDPs by analyzing the features of the situation in which they should be engineered and used; and (5) iterative-incremental engineering and transition of SDMs. Despite these advantages, the method suffers from the following weaknesses: (1) complexity due to the multitude of steps involved (compared with existing SME methods); (2) inability to provide a standard language for describing the output products; (3) lack of a standard tool that supports the SME process; and (4) inability to support short one-day development and operations cycles as prescribed by the DevOps framework. It should also be noted that the proposed method is expected to be applicable for engineering various SDPs in various situations; however, we could not empirically test it in larger organizations, so this feature remains unexplored.

The results of this research can help practitioners engineer SME methods and SDMs that support the KM process. Researchers can also reuse the constituents of the proposed method to improve in-use SME methods. We aim to further this research by proposing a new version of our proposed SME method that supports ontology-based description of SDMs. Furthermore, we plan to provide a refined version of the proposed SME method that supports one-day DevOps cycles. Another strand of research will focus on conducting further case studies in different contexts; this will provide a chance to improve the proposed SME method through further iterations of the Assess/Refine cycle of the research methodology (Fig. 2).

References

1. Ramsin R, Paige RF (2008) Process-centered review of object oriented software development methodologies. ACM Comput Surv 40:1–89. https://doi.org/10.1145/1322432.1322435
2. Henderson-Sellers B, Ralyté J, Agerfalk PJ, Rossi M (2014) Situational method engineering. Springer, Berlin, Heidelberg
3. Alavi M, Leidner DE (2001) Knowledge management and knowledge management systems: conceptual foundations and research issues. MIS Q 25:107–136. https://doi.org/10.2307/3250961
4. Jabbari R, bin Ali N, Petersen K, Tanveer B (2016) What is devops?: A systematic mapping study on definitions and practices. In Proceedings of the Scientific Workshop Proceedings of XP 2016. ACM, pp 1–11
5. Httermann M (2012) DevOps for developers. Apress, New York
6. Osterweil L (2011) Software processes are software too. Engineering of Software. Springer, pp 323–344. https://doi.org/10.1145/253228.253440
7. Durst S, Zieba M (2019) Mapping knowledge risks: towards a better understanding of knowledge management. Knowl Manag Res Pract 17:1–13. https://doi.org/10.1080/14778238.2018.1538603
8. Durst S, Zieba M (2021) A knowledge management-driven and devops-based method for situational method engineering: supplementary file. Figshare. Journal contribution. https://figshare.com/s/074895e29090a831c8c4. Accessed 28 October 2021
9. Nehan YR, Deneckere R (2007) Component-based situational methods, In Working Conference on Method Engineering. Springer, Boston. pp 161–175
10. Ramsin R (2006) Engineering of an object-oriented software development methodology. Dissertation, University of York
11. Mirbel I, Ralyté J (2006) Situational method engineering: combining assembly-based and roadmap-driven approaches. Requir Eng 11:58–78. https://doi.org/10.1007/s00766-005-0019-0
12. Aharoni A, Reinhartz-Berger I (2008) A domain engineering approach for situational method engineering. In: Li Q, Spaccapietra S, Yu E, Olivé A (eds) Conceptual modelling–lecture notes in computer science. Springer, Berlin, Heidelberg, pp 455–468
13. Serour MK, Henderson Sellers, B (2004) Introducing agility: a case study of situational method engineering using the OPEN process framework. In: 28th annual international computer software and applications conference (COMPSAC). IEEE, pp 50–57
14. Seidita V, Cossentino M, Gaglio S (2007) Adapting passi to support a goal oriented approach: a situational method engineering experiment. In: Fifth European Workshop on Multi-Agent Systems (EUMAS'07). pp 1–15
15. Bajec M, Vavpotič D, Krisper M (2007) Practice-driven approach for creating project-specific software development methods. Inf Softw Technol 49:345–365. https://doi.org/10.1016/j.infsof.2006.05.007
16. Hoppenbrouwers S, van Bommel P, Järvinen A (2008) Method engineering as game design: an emerging HCI perspective on methods and CASE tools. In: Proceedings of EMMSAD. Citeseer, pp 1–15
17. Kornyshova E, Deneckère R, Salinesi C (2007) Method chunks selection by multicriteria techniques: an extension of the assembly-based approach. In: Working conference on method engineering. Springer, pp 64–78
18. Karagiannis D, Burzynski P, Utz W, Buchmann RA (2019) A metamodeling approach to support the engineering of modeling method requirements. In: 27th international requirements engineering conference (RE). IEEE, pp 199–210
19. Janković M, Žitnik S, Bajec M (2019) Reconstructing de facto software development methods. Comput Sci Inf Syst 16:75–104. https://doi.org/10.2298/csis180226038j
20. Sandkuhl K, Seigerroth U (2019) Method engineering in information systems analysis and design: a balanced scorecard approach for method improvement. Softw Syst Model 18:1833–1857. https://doi.org/10.1007/s10270-018-0692-3
21. Franch X, Ralyté J, Perini A, Abelló A, Ameller D, Gorroñogoitia J, Nadal S, Oriol M, Seyff N, Siena A, Susi A (2018) A situational approach for the definition and tailoring of a data-driven software evolution method. In: Krogstie J, Reijers HA (eds) Advanced information systems engineering. Springer International Publishing, Cham, pp 603–618. https://doi.org/10.1007/978-3-319-91563-0_37

22. Mishra D, Aydin S, Mishra A, Ostrovska S (2018) Knowledge management in requirement elicitation: Situational methods view. Comput Stand Interfac 56:49–61. https://doi.org/10.1016/j.csi.2017.09.004
23. Park JG, Lee J (2014) Knowledge sharing in information systems development projects: explicating the role of dependence and trust. Int J Project Manage 32:153–165. https://doi.org/10.1016/j.ijproman.2013.02.004
24. Pee LG, Kankanhalli A, Kim HW (2010) Knowledge sharing in information systems development: a social interdependence perspective. J Assoc Inform Syst 11:550–575
25. López L, Costal D, Ralyté J, Franch X, Méndez L, Annosi MC (2016) OSSAP–a situational method for defining open source software adoption processes. In: International conference on advanced information systems engineering. Springer, pp 524–539. https://doi.org/10.1007/978-3-319-39696-5_32
26. Mitchell SM, Seaman CB (2016) Could removal of project-level knowledge flow obstacles contribute to software process improvement? A study of software engineer perceptions. Inf Softw Technol 72:151–170. https://doi.org/10.1016/j.infsof.2015.12.007
27. Unterkalmsteiner M, Gorschek T, Islam AM, Cheng CK, Permadi RB, Feldt R (2011) Evaluation and measurement of software process improvement—a systematic literature review. IEEE Trans Software Eng 38:398–424. https://doi.org/10.1109/tse.2011.26
28. Lee MC, Chang T (2006) Applying TQM, CMM and ISO 9001 in knowledge management for software development process improvement. Int J Serv Stand 2:101–115. https://doi.org/10.1504/ijss.2006.008161
29. Almomani MA, Basri S, Kamil A (2018) Factors influencing the implementation of software process improvement in small and medium enterprises: an empirical study. Adv Sci Lett 24:1716–1722. https://doi.org/10.1166/asl.2018.11144
30. Sun Y, Liu XF (2010) Business-oriented software process improvement based on CMMI using QFD. Inf Softw Technol 52:79–91. https://doi.org/10.1016/j.infsof.2009.08.003
31. Basili VR, Caldiera G, Rombach HD (2002) Experience factory. Encycl Software Eng 1:1–19. https://doi.org/10.1002/0471028959.sof110
32. Basili VR (1993) The experience factory and its relationship to other improvement paradigms. In: Sommerville I, Paul M (eds) Software engineering-lecture notes in computer science. Springer, Berlin, Heidelberg, pp 1–14
33. Hevner AR, March ST, Park J, Ram S (2008) Design science in information systems research. Manag Inf Syst Q 28:9–22. https://doi.org/10.1007/978-1-4419-5653-8_2
34. Clarke P, O'Connor RV (2012) The situational factors that affect the software development process: towards a comprehensive reference framework. Inf Softw Technol 54:433–447. https://doi.org/10.1016/j.infsof.2011.12.003
35. Khatibian N, Hasan gholoi pourAbedi Jafari TH (2010) Measurement of knowledge management maturity level within organizations. Bus Strateg Ser 11:54–70. https://doi.org/10.1108/17515631011013113
36. Henderson-Sellers B (2005) Agent-oriented methodologies. IGP, US and UK
37. Silva A, Araújo T, Nunes J, Perkusich M, Dilorenzo E, Almeida H, Perkusich A (2017) A systematic review on the use of definition of done on agile software development projects. In: 21st international conference on evaluation and assessment in software engineering. ACM. pp 364–373
38. d Santos FrançaNettodo Carvalho JBJMES et al (2015) KIPO: the knowledge-intensive process ontology. Softw Syst Model 14:1127–1157. https://doi.org/10.1007/s10270-014-0397-1
39. Object Management Group (OMG) (2008) About the software and systems process engineering metamodel specification version 2.0. https://www.omg.org/spec/SPEM/About-SPEM. Accessed 17 September 2020
40. Riege A (2005) Three-dozen knowledge-sharing barriers managers must consider. J Knowl Manag 9:18–35. https://doi.org/10.1108/13673270510602746
41. Balachandran V (2020) On the need for automatic knowledge management in modern collaboration tools to improve software maintenance. In: IEEE international conference on software maintenance and evolution (ICSME). IEEE, pp 722–722
42. Duffield SM, Whitty SJ (2016) Application of the systemic lessons learned knowledge model for organisational learning through projects. Int J Project Manage 34:1280–1293. https://doi.org/10.1016/j.ijproman.2016.07.001
43. Shakeri Hossein Abad Z, Gervasi V, Zowghi D, Far BH (2019) Supporting analysts by dynamic extraction and classification of requirements-related knowledge. In: 41st international conference on software engineering (ICSE). IEEE/ACM, pp 442–453
44. Ali N, Daneth H, Hong JE (2020) A hybrid DevOps process supporting software reuse: a pilot project. J Softw Evol Process 32:1–23. https://doi.org/10.1002/smr.2248
45. Li Y, Tarafdar M, Rao SS (2012) Collaborative knowledge management practices. Int J Oper Prod Manag 32:398–422. https://doi.org/10.1108/01443571211223077
46. López-Martínez J, Juárez-Ramírez R, Huertas C, Jiménez S, Guerra-García C (2016) Problems in the adoption of agile-Scrum methodologies: a systematic literature review. In: 4th international conference in software engineering research and innovation. IEEE, pp 141–148
47. Haumer P (2007) Eclipse process framework composer. https://www.eclipse.org/epf/general/EPFComposerOverviewPart1.pdf. Accessed 20 September 2020
48. Brereton P, Kitchenham B, Budgen D, Li Z (2008) Using a protocol template for case study planning. In: 12th International conference on evaluation and assessment in software engineering (EASE). ScienceOpen, pp 1–8
49. Kitchenham B, Al-Khilidar H, Babar MA, Berry M, Cox K, Keung J, Kurniawati F, Staples M, Zhang H, Zhu L (2008) Evaluating guidelines for reporting empirical software engineering studies. Empir Softw Eng 13:97–121. https://doi.org/10.1007/s10664-007-9053-5
50. Jedlitschka A, Pfahl D (2005) Reporting guidelines for controlled experiments in software engineering. In: International symposium on empirical software engineering. IEEE, pp 95–104
51. Runeson P, Host M, Rainer A, Regnell B (2012) Case study research in software engineering: guidelines and examples. Wiley, Hoboken, New Jeresy and Canada
52. Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2012) Experimentation in software engineering. Springer, Berlin. https://doi.org/10.1007/978-3-642-29044-2
53. Mohapatra S, Bose S (2020) An appraisal of literature for design and implementation of developing a framework for digital twin and validation through case studies. Health Technol 10:1229–1237. https://doi.org/10.1007/s12553-020-00443-4
54. Banerjee SS, Mohapatra S, Saha G (2021) Developing a framework of artificial intelligence for fashion forecasting and validating with a case study. Int J Enterp Netw Manag 12:165–180. https://doi.org/10.1504/IJENM.2021.10039608
55. Mohapatra S (2019) Critical review of literature and development of a framework for application of artificial intelligence in business. Int J Enterp Netw Manag 10:176–185. https://doi.org/10.3390/jrfm14120604
56. Rubin KS (2012) Essential Scrum: a practical guide to the most popular agile process. Addison-Wesley, US
57. Brown JD (2011) Likert items and scales of measurement. Statistics 15: 10–14. http://hosted.jalt.org/test/bro_34.htm

58. Thompson MP, Walsham G (2004) Placing knowledge management in context. J Manage Stud 41:725–747. https://doi.org/10.1111/j.1467-6486.2004.00451.x
59. Podsakoff PM, MacKenzie SB, Lee JY, Podsakoff NP (2003) Common method biases in behavioral research: a critical review of the literature and recommended remedies. J Appl Psychol 88:879–903. https://doi.org/10.1037/0021-9010.88.5.879
60. Armstrong JS, Overton TS (1977) Estimating nonresponse bias in mail surveys. J Mark Res 14:396–402. https://doi.org/10.2307/3150783