Journal of Applied Research and Technology

www.jart.ccadet.unam.mx
Journal of Applied Research and Technology 17 (2019) 28-43

Original

Bottom-up authoring of software engineering methods and practices

Miguel Ehécatl Morales-Trujillo [a, *], Hanna Oktaba [b], Mario Piattini [c],
Boris Escalante-Ramírez [a]

[a] *Facultad de Ingeniería, Universidad Nacional Autónoma de México, Mexico City, Mexico*

[b] *Facultad de Ciencias, Universidad Nacional Autónoma de México, Mexico City, Mexico*
Av. Universidad 3000, Ciudad Universitaria, 04510, Mexico City, Mexico

[c] *Alarcos Research Group, University of Castilla – La Mancha,*
Paseo de la Universidad 4, 13071, Ciudad Real, Spain

Received dd mm aaaa; accepted dd mm aaaa
Available online dd mm aaaa

Abstract: Software Engineering knowledge is obtained during software engineering efforts, such as projects, experiments and case studies that represent a valuable source of knowledge with which to enrich the discipline. This knowledge is manipulated by practitioners who are in charge of developing, maintaining or integrating software; any practitioner, experienced or beginner, possesses his/her tacit practices in order to carry out their work. However, these ways of working are frequently neither expressed nor collected in order to reason about their characteristics and properties. Moreover, the explicit ways of working, which are presented in process reference models and standards that follow a prescriptive approach, are not suitable for small software development organizations. Small organizations represent a major part of software development organizations, so it is important to know and support how they actually work.
This paper describes KUALI-BEH, a bottom-up metamodel that offers software engineering practitioners an authoring framework with which to express, adapt and share their ways of working as a collection of methods and practices. Validating and applying the metamodel showed that a bottom-up approach benefits small organizations and serves as a first step to reduce the gap between software engineering theory and practice. KUALI-BEH permits small organizations to create an organizational working method and to gradually introduce them to the adoption of standards and reference models. Practitioners with different levels of competence, from inexperienced to senior, adopt and use KUALI-BEH successfully with minimal training and without any consulting services.

Keywords: Practice, method, software engineering, bottom-up, OMG, KUALI-BEH, VSE.

1. INTRODUCTION

Software Engineering is part of an engineering discipline and must be based on scientific practices and theory in order to justify their approaches and to provide scientific evidence as to why and where their methods work properly (Broy, 2011). However, Software Engineering is currently controlled and guided by the practices used in industry and technological innovations (Wang, 2008), while the theory and foundations of Software Engineering, which are essential in supporting its practices, are left aside (Jacobson, Johnson, & Ekstedt, 2012).

In an attempt to address these concerns, the software engineering community has created and promoted a considerable number of proposals. We can mention such

[*] Corresponding author.
E-mail address: migmor@ciencias.unam.mx (Miguel Ehécatl Morales-Trujillo).
Peer Review under the responsibility of Universidad Nacional Autónoma de México.

metamodels as SPEM (OMG, 2008) or SEMDM (ISO, 2007); Method Engineering (Brinkkemper, 1996) and Situational Method Engineering (Harmsen, Brinkkemper, & Oei, 1994), which were conceived with the aim of building methods from pre-existing pieces of methods; a collection of ISO/IEC Software Engineering standards; and efforts like Software Engineering Method and Theory[1] (SEMAT), whose objective is to concentrate and define the theory needed to re-found the discipline providing mechanisms to express methods and practices using a common language.

Nevertheless, the aforementioned proposals benefit large organizations to a greater extend, becoming an obstacle for smaller ones that do not follow models or standards or have low levels of maturity. As reported by (Oktaba & Piattini, 2008), 58% of organizations working on significant software products are small organizations, and according to (Basri & O'Connor, 2010) 85% of IT organizations have 1 to 10 employees. When a very small entitiy (VSE), less than 25 employees, adopts a new standard, it simultaneously needs to change its structure, culture, organizational climate and projects, which, in its turn, leads to changes in its ways of working. In fact, small organizations actually abandon what they were doing in order to adapt new models, while the cost of the adoption is disproportionate to its benefit (Espinosa-Curiel, Rodríguez-Jacobo, & Fernández-Zepeda, 2016).

Moreover, there is evidence that, although the existing standards do contain good practices, the majority of small software organizations do not adopt them because they are perceived to be oriented towards large organizations (Laporte, Alexandre, & O'Connor, 2008). For example, in (Staples et al., 2007) the authors state that small organizations consider themselves too small to adopt CMMI (SEI, 2010). According to (Pino, Pardo, García, & Piattini, 2010), these standards per se are not suitable for small organizations. In fact, size is one of the main reasons to avoid the adoption of standards; the others are cost and effort. In addition, Coleman & O'Connor (2008) establish that, although commercial software process improvement (SPI) models have been highly publicized and marketed, they are not being widely adopted and their influence in the software industry therefore remains more at a theoretical than practical level. These concerns reinforce the need for strategies for process improvement that are tailored to small companies' characteristics (Pino et al., 2010).

In practice, small organizations predominantly acquire knowledge through observations, which are obtained during software engineering efforts. Practitioners then apply these observations to their activities, therefore constantly in need of an effective thinking framework that will bridge the gap between their current ways of working and any new ideas that they wish to adopt within their organizations (Jacobson, Ng, McMahon, Spence, & Lidman, 2012). Such framework should be able to help them discover the hidden but valuable knowledge they possess while respecting their work culture.

This paper presents a bottom-up metamodel, KUALI-BEH (OMG, 2012), which was created specifically for small organizations with the aim of responding to the above-mentioned need. KUALI-BEH pursues the objective of becoming a starting point between practice and theory in Software Engineering, based on a bottom-up approach with its focus being on practitioners; specifically, it guides them in the task of expressing their ways of working in the form of practices and methods. This authoring framework allows practitioners to reason about inherent characteristics and properties of their daily activities, which is a first step towards both improvement and incorporation of practitioners into the theoretical knowledge related to the Software Engineering discipline.

Additionally, it permits small organizations to create an organizational repository comprising their knowledge and to gradually introduce them to the adoption of standards and reference models. Once this stored knowledge has been validated and approved by the software engineering community, it is likely to become a collection of useful material with educational purposes for software engineering courses.

This paper is organized as follows: Section 2 presents the related and prior work surrounding KUALI-BEH, while Section 3 expands on the KUALI-BEH Software Project Common Concepts metamodel. The bottom-up authoring framework is presented in Section 4. The conclusions and future work are provided in Section 5.

2. BACKGROUND: AN ANALYSIS OF THE CURRENT SITUATION

This section shows the general state-of-the-art surrounding KUALI-BEH.

2.1 SOFTWARE PROCESS MODELS

The importance of creating software process descriptions with which to guide key software processes was highlighted by Osterweil in the well-known paper *Software processes are software too* (Osterweil, 1987) in the late 80's. Later, from the 90's on, there has been an increasing interest in developing proposals for process modelling. Among the various proposals that exist, there are several metamodels with similar purposes, such as Process Interchange Format (PIF) (Lee et al., 1998) and Process Specification Language (PSL) (Schlenoff, Knutilla, & Ray, 1996), whose objective is to provide a standard exchange format in order to share process models between organizations. Other existing metamodels are Entry-Task-Validation-Exit method (ETVX) (Radice, Harding, Munnis, & Phillips, 1985), Integration Definition for Function Modelling (IDEF) (KBSI, 1993), Core Plan Presentation (CPR) (Pease & Carrico, 1997), Shared Planning and Activity Representation (SPAR) (Tate, 1998), PROMENADE (Franch & Ribó, 1999) and SPEARMINT (Becker-Kornstaedt, Scott, & Zettel, 2000).

There are consequently many proposals with which to model software processes, and this implies a high degree of heterogeneity that requires greater standardization. With that goal in mind, two proposals (described below) can be highlighted.

The first proposal under consideration is the Software and Systems Process Engineering Meta-model (SPEM) (OMG, 2008), defined in 2008 by the Object Management Group (OMG). SPEM is both a framework and an engineering process metamodel with the objective to provide necessary concepts and language for representing, managing and sharing methods and processes.

Albeit its benefits, SPEM's drawbacks are related to the semiformal architecture that makes the verification of the statement created extremely difficult (Krdžavac, Gašević, & Devedžić, 2009), and to SPEM's primary focus on the definition of a process, while aspects of its execution such as sequence and the order of the method components are sidelined.

The second proposal under consideration is the ISO/IEC 24744 standard (ISO, 2007), aka the Software Engineering – Metamodel for Development Methodologies (SEMDM), which provides a framework based on vocabulary and semantics for the definition and extension of methodologies in information-based domains, such as software development methodologies. The method engineers in charge of methodologies definition are the target audience of SEMDM, in an attempt to facilitate the communication between them and practitioners. SEMDM includes three main aspects: the process to follow, the products used and generated and the people involved.
Despite the advantages of the formalism and structure of the SEMDM metamodeling approach, the proposed concepts are not familiar to practitioners and are distanced from their context. For example, the term *clabject* mixes the idea of class and object, leading to confusion and a lack of use among practitioners.
Finally, SEMDM and SPEM are top-down approaches, which are principally focused on method engineers and prescribe practitioners concerning how to perform their activities. In contrast, the bottom-up approach of KUALI-BEH promotes the description of practitioners' actual ways of working and encourages synergy between practitioners and method engineers.

2.2 PROCESS REFERENCE MODELS AND STANDARDS

Process reference models or standards, such as CMMI, ISO/IEC 12207 (ISO, 2008a), ISO/IEC 15504 (ISO, 2004), and ISO/IEC 29110 (ISO, 2012); contain validated and approved knowledge to be applied by practitioners. However, this knowledge may undergo transformations when in use. According to (Osterweil, 1987) a static process description is constructed to specify a collection of dynamic processes. A process may consequently be performed in many ways, but not all of them correctly. These models operate primarily at a theoretical level; and in the words of (Coleman & O'Connor, 2007) are too prescriptive and bureaucratic to implement in practice. Besides they require a subscribing company to adapt to the models rather than having the models easily adapt to them.

There are other issues that need to be addressed. According to (Clarke et al., 2016a), software process domain suffers from an inconsistent use of terminology. Homogeneity between standards has not yet been achieved, and in (Rout, 1999) the need to align the large number of definitions stated in standards is reinforced, particularly in SC7 in which 198 of the 864 definitions were duplicated or contained overlapping terms (Henderson-Sellers, Gonzalez-Perez, McBride, & Low, 2014).

There is a large, complex and potentially very costly problem concerning the present application of terminology to both processes and roles involved in software development (Clarke et al., 2016b). It is therefore crucial to enclose and express the process instances in a homogeneous manner, in order to analyze them and decide which are valid and which are not.

All of the above makes the transition complex for both practitioners and the organization, thus increasing the effort required to understand, adopt and train into the new model, which is a time- and cost-consuming task, particularly for VSEs.

On the one hand, the top-down strategy of standard adoption in organizations forces practitioners to adapt or replace their actual ways of working in order to conform to the standard or model correctly. The prescriptive nature of process reference models and standards lessens the expression of their acquired knowledge and halts their actual practices. What is more, there is a growing concern in the software engineering community that SEI and ISO standards are not easily applicable to small firms because they require a huge investment (in time, money, and resources) (Pino, García, & Piattini, 2008). Besides, even if managers were familiar with CMMI or ISO 9000 they were against introducing it to their new organizations (Coleman & O'Connor, 2007).

On the other hand, (Staples et al., 2007) consider that SPI approaches must target small software-developing organizations and should require very little cost and time to adopt. One of the ways to achieve this is to base SPI on a bottom-up approach. Its use would enable practitioners to author the way they actually carry out activities, thus reducing the aforementioned difficulties. This approach is discussed in more detail is section 4.1.

2.3 KNOWLEDGE MANAGEMENT IN VSEs

Human beings gain expertise and transform it into knowledge through perception, intuition and experience, rather than by following a predefined process (Dreyfus & Dreyfus, 1986). As for Software Engineering, knowledge and its management become particularly relevant since software development projects involve intensive knowledge exchanges and collaborations (Basri & O'Connor, 2011). Therefore, knowledge needs to be properly managed to ensure that the right knowledge gets into the right place (O'Connor & Basri, 2014).

In VSEs, adequate knowledge management can be one of the factors to increase their innovation power (O'Connor & Basri, 2014) and creativity (Edwards, 2003). However, (Basri & O'Connor, 2011) raise concerns about the performance of knowledge management related activities in VSEs; in fact, the authors found out that knowledge management in VSEs gained less than satisfied agreement level. Besides, there is an absence of published material describing how process is initially formed in software product companies (Coleman & O'Connor, 2007), so even if organizations do want to take advantage of the knowledge gained through experience, they lack published references to guide themselves in this task.

On the other hand, companies who use tacit knowledge extensively recognize that it has its limitations and may ultimately carry its own cost (Coleman & O'Connor, 2008). O'Connor & Basri (2014) report that learning and sharing processes in VSEs are weak due to informal communication, informal documentation and autonomous work habits, which, therefore, need to be improved. Nevertheless, the same authors consider that well-organized software engineering knowledge may assist VSEs in maintaining their product relevancy in the market, among other aspects. A first step towards this improvement is to transform the tacit knowledge into explicit in an organized and systematical manner.

A useful and effective alternative to achieve this is by using a common terminology. Through the use of common terms with clear-cut definitions to represent the tacit knowledge of VSE will allow to bridge the gap from an idea to its representation (Clarke et al., 2016a).

Besides, this transformation of knowledge should follow well-defined steps in order to be productive. Edwards (2000) proposes a model that describes what happens to a particular element of knowledge from an organizational viewpoint (see Figure 1). First, the organization creates or acquires a new piece of knowledge, then it goes through Retain, Use and Refine/Update cycle, where its goal is to be used and get improved. It may also be Shared/Transferred to other organizations or entities; and the activities of Retaining-Using-Refining and Sharing may co-occur.

Based on this process, it is feasible to define a process for managing software engineering knowledge in VSEs, which will allow them to explicitly express the knowledge they possess. The whole purpose of KUALI-BEH, which is

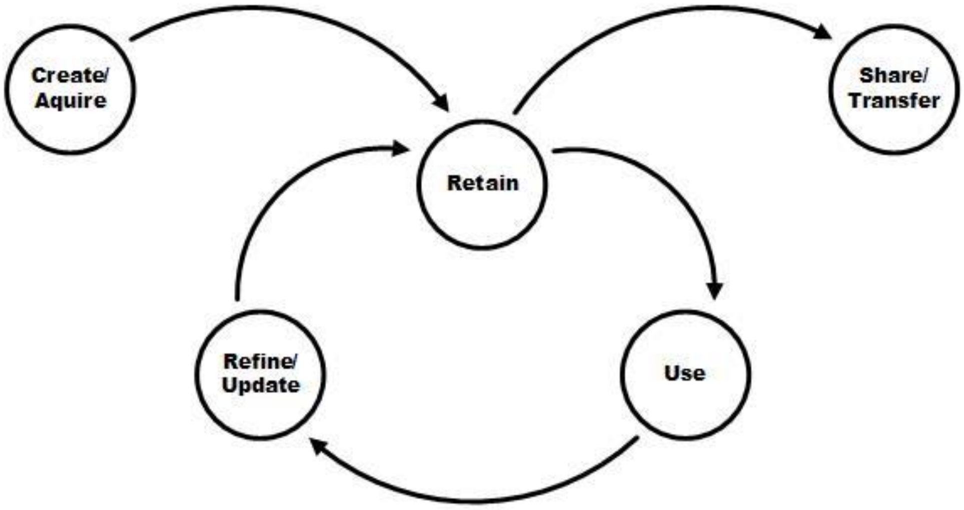

Fig. 1. Knowledge management process proposed by Edwards, 2000.

presented in the next section, is to assist practitioners in this task, and section 4.2 describes a set of states that take in consideration the steps proposed in (Edwards, 2000).

3. KUALI-BEH SOFTWARE COMMON CONCEPTS

In order to reduce the software development process terminological challenge, the concept orientation and the systematic terminology work approach are key (Clarke et al., 2016a). Following this idea, KUALI-BEH, the framework under consideration, describes concepts that are common for all software projects and are necessary in order to express how the practitioners work, i.e. to define their ways of working. Thus, being formally expressed as practices and arranged into methods, they make an organizational base of knowledge that can be applied by work teams during software projects. The knowledge produced by practitioners needs validation and approval before being accumulated and shared, both inside and outside the organization.

The research method used to create KUALI-BEH was a combination of Technical-Action-Research (TAR) (Wieringa & Morali, 2012; Wieringa, 2014) and case studies (Runeson, Host, Rainer, & Regnell, 2012). In TAR, an artifact is designed to solve a particular problem and is immersed in a real context where it is tailored and improved. The immersion and improvement activities constitute engineering cycles, through which the artefact is moving until it becomes an effective solution for the problem.

In particular, the first engineering cycle, which gave birth to the initial version of KUALI-BEH, was a response to an RFP for a Foundation for the Agile Creation and Enactment of Software Engineering Models (FACESEM) (OMG, 2011) launched by the OMG. KUALI-BEH followed the requirements defined by FACESEM and gathered the key concepts involved in software projects and needed in order to collect practitioners' knowledge in terms of practices and methods.

To create the initial set of common concepts, we started by collecting knowledge from recognized sources (ISO, 2004; ISO, 2005; ISO, 2007; ISO, 2008a; ISO, 2008b; ISO 2010; ISO, 2012; OMG, 2008; PMI, 2008; Rout, 1999; SEI, 2010; Schwaber & Sutherland, 2017) and taking advantage of our experience in defining software development standards (ISO, 2012; NYCE 2011; Oktaba, et al., 2007; OMG, 2014).

The terms found in the above mentioned standards were collected, compared and harmonized. In this cycle, the number of common concepts was narrowed down to 20 and was represented as an ontology, which helped to interrelate concepts and their definitions (see Table 1). For the full representation, refer to (OMG, 2012).

According to (Wang, Wang, Zhuang, & Fei, 2015) an ontology, as a shared conceptualization, expresses a consensus among people; and an ontological framework can help to ground the language usage in a field (Clarke et al., 2016a). For the definitions to be less formal and more understandable for practitioners, dictionaries and thesaurus of the English language were consulted, and a better and more easily understandable set of concepts was achieved.

Once the first version was developed, a proof of concept was carried out where agile and traditional practices, Scrum and ISO/IEC 29110, were expressed through KUALI-BEH. Subsequently, the proposal went through five engineering cycles, was validated in three case studies, reviewed by experts from the OMG task forces, and, finally, was formalized by means of Description Logics (Morales-Trujillo, Oktaba, Hernández-Quiroz, & Escalante-Ramírez, 2018). For graphic representation of the whole process, see Figure 2. The validation process is described in more detail in section 4.3.

Nowadays KUALI-BEH is part of the ESSENCE – Kernel and Language for Software Engineering Methods (OMG, 2014) standard. The contribution of the framework to the standard consists in providing an improved and refined definition of method and practice concepts, a formal definition of method properties, operations of adaptation for method tailoring and a board-based approach for project management in small organizations.

KUALI-BEH is primarily oriented toward software engineering practitioners. Its bottom-up approach relies on the premise that practitioners have been developing their work without process reference models, standards or specifications with an acceptable rate of success.

Table. 1. Fragment of the KUALI-BEH common concepts ontology (OMG, 2012).

Name	Definition	Example
Activity	An activity is a set of tasks that contributes to the achievement of a practice objective.	SI.2.2 Document or update the Requirements Specification.
Condition	A condition is a specific situation, circumstance or state of something or someone with regard to appearance, fitness or working order that have a bearing on the software project.	The team is working together and every member of the team is in context for the coming day's work.
Knowledge and Skills	The knowledge and skills are a set of abilities, competences and attainments, acquired by the practitioner and needed to perform a practice.	- Experience eliciting requirements - Experience in designing user interfaces - Knowledge of the revision techniques.
Practitioner	A practitioner is a professional in Software Engineering that is actively engaged in the discipline. The practitioner should have the ability to make a judgment based on his or her experience and knowledge.	Hanna, Miguel
Project Conditions	The project conditions are the factors related to the project that could affect its realization. Complexity, size, time and financial restrictions, effort, cost and other factors of the project environment are considered. It is a specialization of a condition.	KB-Project-Conditions

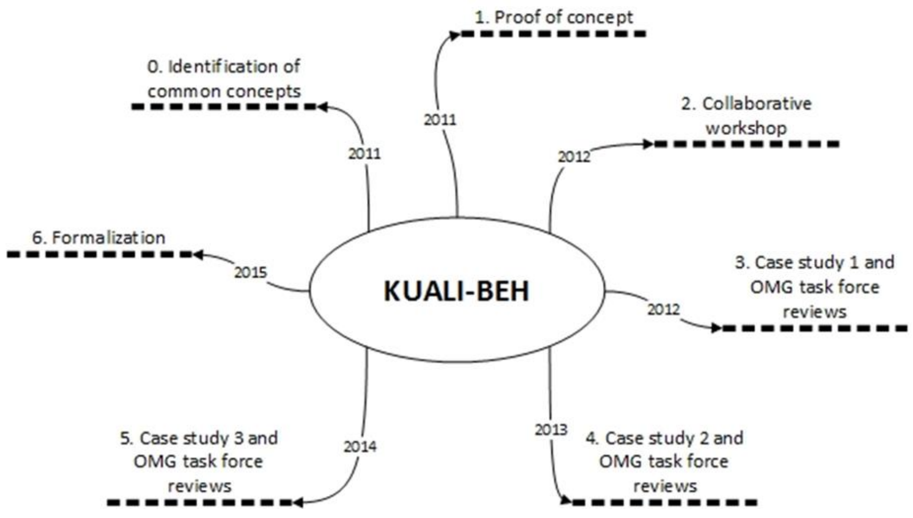
1. Proof of concept
0. Identification of common concepts
2011
2011
2. Collaborative workshop
2012
6. Formalization
2015
KUALI-BEH
3. Case study 1 and OMG task force reviews
2012
5. Case study 3 and OMG task force reviews
2014
2013
4. Case study 2 and OMG task force reviews

Fig. 2. KUALI-BEH engineering cycles.

3.1 KUALI-BEH

According to KUALI-BEH (OMG, 2012), software project common concepts (written in italics) can be outlined as follows:

A *software project* is the effort of a group of software engineering *practitioners* whose objective is to develop, maintain or integrate *software products*. A software project typically originates from the needs of an individual or an organization: a *stakeholder*. The *stakeholder needs* are expressed to a *work team*, composed of practitioners, under certain restrictions called *project conditions*.

While work teams are developing software projects, they are creating their own ways of working consisting of *methods* and *practices* according to their own *knowledge and skills*. A practice is, therefore, a set of *activities* and *tasks,* which has been repeatedly used in software projects and has proved to be useful. For a graphical representation of the concepts, see Figure 3.

The methods and practices can be structured as a base of knowledge aimed at collecting and concentrating the existing ways of working in the form of units, so they can be consulted and analyzed by the work team in order to select the appropriate method responding to the particular context of a software project. Another goal of collecting method and practices is to foster the modification of practices and methods in a controlled manner.

These 20 common concepts constitute the vocabulary proposed by KUALI-BEH in order to represent the tacit knowledge that practitioners possess. KUALI-BEH common concepts are intended to be the means of description, analysis and reasoning about software projects and the information related to them.

During the authoring of methods and practices, KUALI-BEH advises practitioners with regard to certain attributes. The set of practices that form a method needs to maintain the properties of coherency, consistency and sufficiency. Coherency is preserved when the practices that compose a method contribute actively towards reaching the method's purpose. In a consistent set of practices, each practice produces a result that is consumed as an input of another practice. Sufficiency is achieved when the method is coherent, consistent and its purpose is entirely fulfilled. Beside that, practices may undergo adaptation operations of substitution, splitting, combination and concatenation. For their full description and graphic representation, see (OMG, 2014).

Altogether, KUALI-BEH places at practitioners' disposal all the necessary elements to formally express the way they work. It offers templates to guide the description of activities and helps to author practices and methods through operations and properties. As a result, valuable knowledge is collected and stored in a repository for everybody to consult, personalize and use it.

4. BOTTOM-UP PRACTICE AND METHOD AUTHORING

As it was mentioned before, every practitioner possesses valuable knowledge that he or she resorts to during software development efforts. This knowledge could be acquired when an organization specifically adopted a model to guide its projects. However, many small organizations develop software without any renowned model, still coping with their tasks at a high level of performance. We consider that their tacit knowledge, acquired through experience and captured in their ways of working, is the source of successful fulfillment of projects, and it needs to be preserved and shared among organizations.

The objective of this section is to outline advantages of the bottom-up as compared to the top-down approach, especially in the context of VSEs. First, we present the bottom-up approach and highlight its suitability to VSEs. Further, we describe ALPHAs for Practice authoring and for Method authoring, which are subordinate of the Way-of-Working ALPHA as defined in (OMG, 2014), and show how practitioners can carry out authoring processes using the ALPHA concept.

4.1 BOTTOM-UP APPROACH

Traditionally, organizations opt for adopting a top-down approach, where the head of the organization chooses a model or standard to be implemented as a new and routine way of working. Thus, actual activities and tasks are changed or replaced for the ones prescribed by the model until every member of the organization has adopted the new way of work.

Such models do not adapt themselves to the organization, but require the organization to adapt to them. In the words of (Staples et al., 2007), while CMMI admits tailoring, the emphasis is on tailoring "down" by demanding justification for excluding defined aspects of the

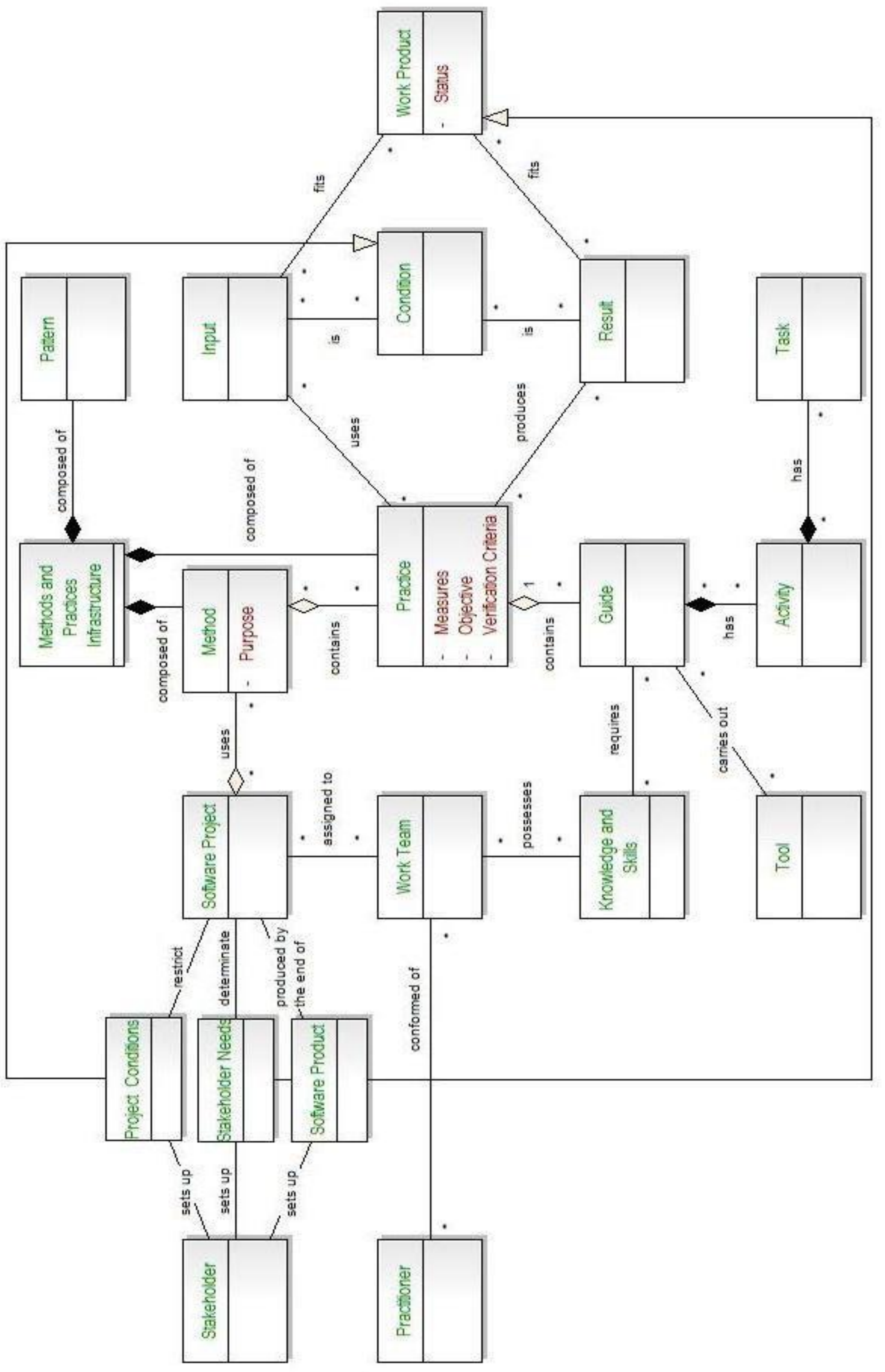

Fig. 3. Software project common concepts class diagram.

model, rather than on tailoring "up" by starting with minimal practices and adding to them as necessary to address specific issues. It is no surprise that a top-down approach causes a partial loss of practitioners' valuable knowledge. In addition, smaller organizations believe that adopting models incurs significant costs and may negatively affect their creativity and flexibility (Coleman & O'Connor, 2008).

When organizations choose a bottom-up approach, the scenario is quite different. In the first place, practitioners who are responsible for expressing and modifying their activities direct it. In the second place, practitioners themselves are in charge of evaluating what they have expressed and of composing their own software engineering method. Altogether, this approach promotes collaboration and communication among members of work teams, involves practitioners into the organization's functioning and reduces resistance to change. While a top-down adoption requires expensive consulting services, a bottom-up approach promotes internal leadership.

What is more, making the knowledge explicit allows for constant improvement. Visibly expressed working practices can be more easily discussed, analyzed and, consequently, improved by the same practitioners.

Finally, a bottom-up approach implies a minimal loss of knowledge; all the knowledge possessed by the organization stays in the organization. Often the knowledge within small organizations is situationally originated and context dependent so its loss becomes undesirable and detriment for organizational well-being. Going from the bottom helps to evaluate, share and preserve it.

Considering these differences in approaching software development endeavors, we chose a bottom-up design for our metamodel (see Figure 4). KUALI-BEH takes into account practitioners' experience and know-how to bring benefit to the organization.

4.2 PRACTICE AND METHOD AUTHORING ALPHAS

In this subsection, we present how KUALI-BEH extends and amplifies the endeavor area of concern with the help of two new ALPHAs: Practice and Method Authoring.

The ALPHA concept is an essential element of a software endeavor. Each ALPHA consists of a set of states that allow practitioners to track the progress of a particular aspect of the endeavor. Each state is defined in terms of a

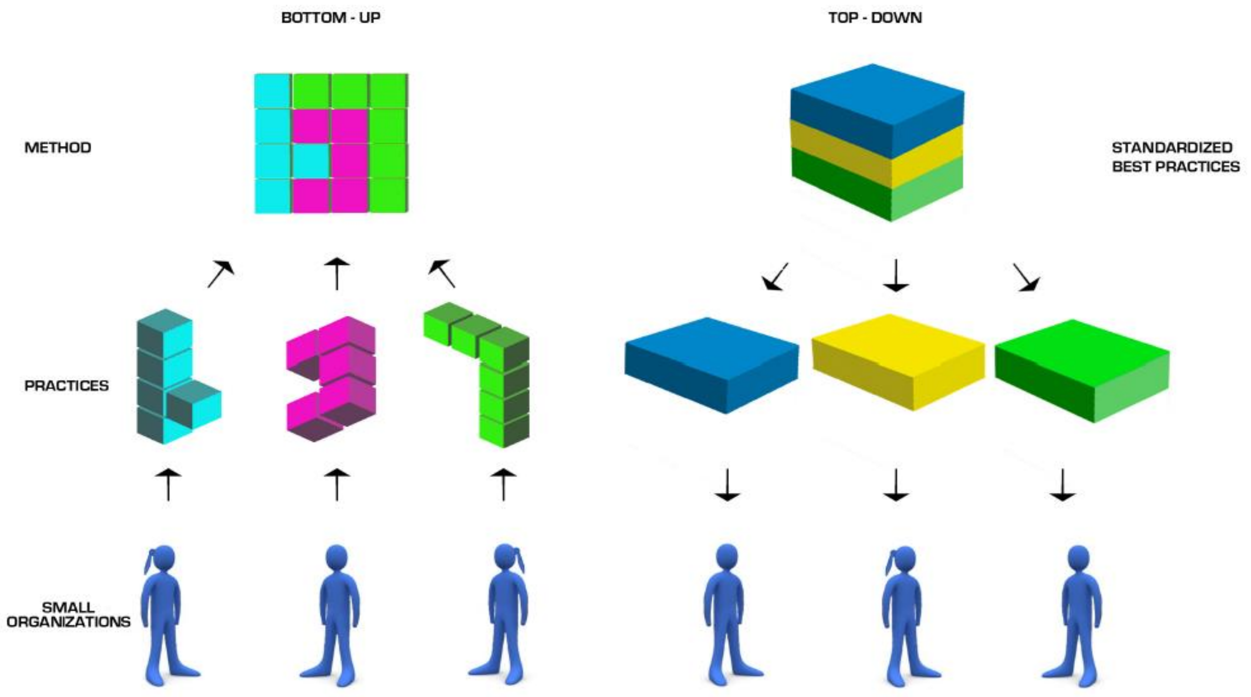

Fig. 4. Bottom-up and top-down approaches.

checklist that helps to understand what to address in order to accomplish an objective. This concept is particularly relevant for VSEs because ALPHAs allow practitioners to transform their tacit knowledge into explicit, expressing their ways of working in terms of practices and later on as methods that emerge in the organizations and belong to them.

The endeavor area of concern embraces the team and its way of working; any method must describe a set of practices to effectively plan, lead and monitor the efforts of the team (OMG, 2014). The Practice Authoring ALPHA is the first step that allows practitioners to express their work units as practices, and the Method Authoring ALPHA is a higher level where these practices are joined to make up methods. The two ALPHAs work together helping practitioners to articulate explicitly their work.

4.2.1 Practice Authoring

The Practice Authoring ALPHA represents a continuum of six states, and the entire authoring process consists in going back and forth from one state to another, i.e. following an iterative process, until the practice reaches its final state: Consolidated. Figure 5 shows the whole set of states for Practice Authoring ALPHA.

In order to determine the state of a practice during authoring, a checklist is provided in (OMG, 2014). For example, if a practice is *Agreed,* according to the checklist, it implies that "the expressed practice has been revised and accustomed by practitioners" and "the expressed practice has been accepted by the practitioners as their explicit way of working".

Due to its simplicity and manageability, the checklist gives the practitioner the necessary information to assess the practice's state and continue the authoring process without additional sources of information or knowledge, e.g. consulting services.

4.2.2 Method Authoring

Method Authoring is an articulation of a coherent, consistent and sufficient set of practices (OMG, 2012). Similar to the Practice Authoring ALPHA, the Method Authoring ALPHA consists of six states, through which a set of practices evolves into a method. Figure 6 displays the Method Authoring ALPHA states with their related definitions. It can be noticed that the second and the third states, *Integrated* and *Well-Formed*, are different from those of the Practice Authoring ALPHA. These two states exploit largely the properties and operations mentioned in 3.1.

All the collected knowledge is stored in a knowledge base and is recycled and taken advantage of in the future software project endeavors.

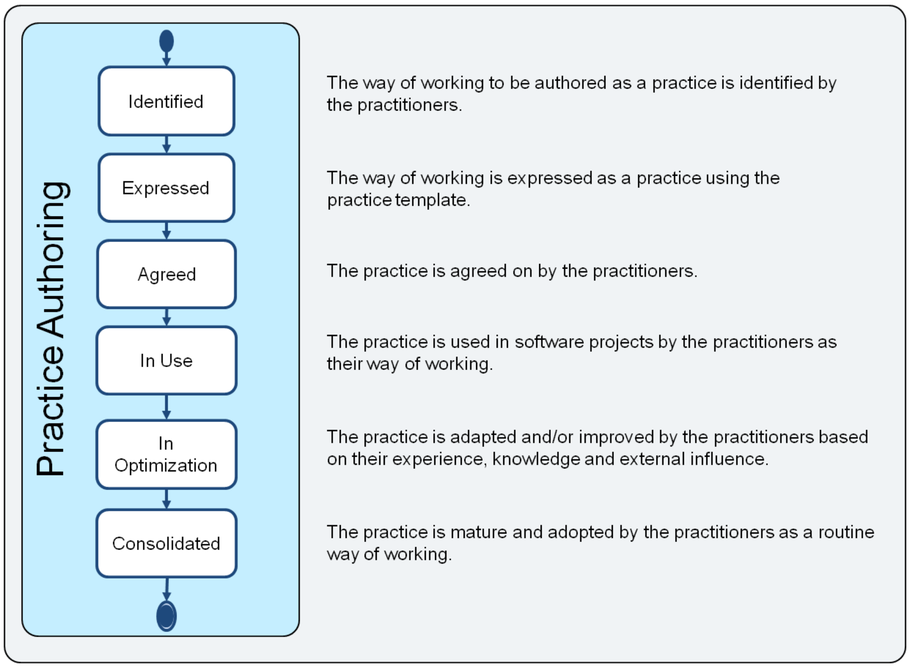
Practice Authoring
Identified
Expressed
Agreed
In Use
In Optimization
Consolidated
The way of working to be authored as a practice is identified by the practitioners.
The way of working is expressed as a practice using the practice template.
The practice is agreed on by the practitioners.
The practice is used in software projects by the practitioners as their way of working.
The practice is adapted and/or improved by the practitioners based on their experience, knowledge and external influence.
The practice is mature and adopted by the practitioners as a routine way of working.

Fig. 5. Practice Authoring ALPHA (OMG, 2014).

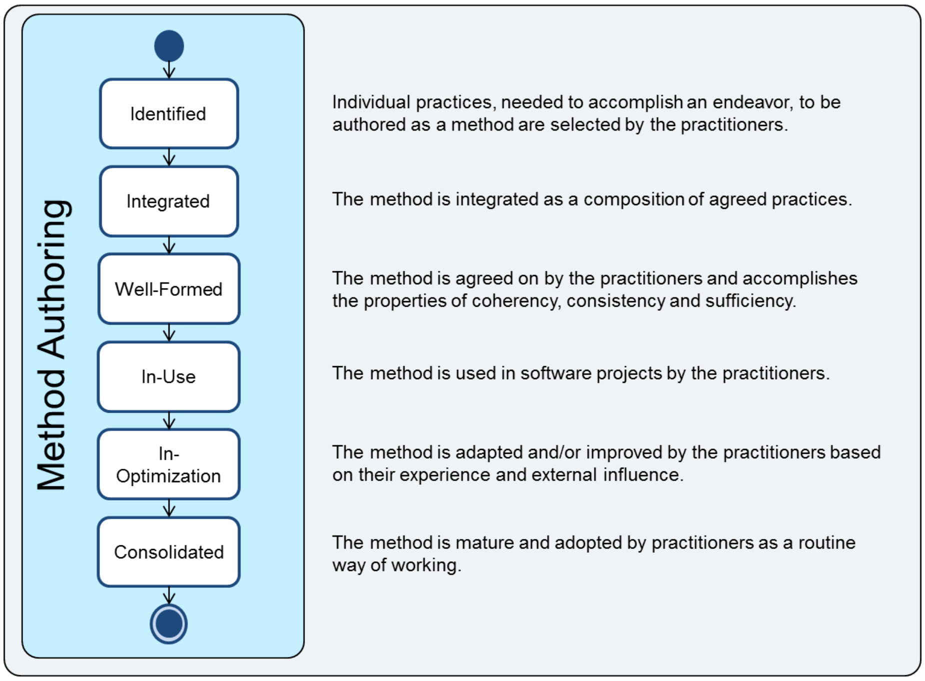
Method Authoring
Identified
Individual practices, needed to accomplish an endeavor, to be authored as a method are selected by the practitioners.
Integrated
The method is integrated as a composition of agreed practices.
Well-Formed
The method is agreed on by the practitioners and accomplishes the properties of coherency, consistency and sufficiency.
In-Use
The method is used in software projects by the practitioners.
In-Optimization
The method is adapted and/or improved by the practitioners based on their experience and external influence.
Consolidated
The method is mature and adopted by practitioners as a routine way of working.

Fig. 6. Method Authoring ALPHA, adapted from (OMG, 2014).

4.3 VALIDATION AND USE IN REAL LIFE CONTEXTS

The KUALI-BEH framework went through a long validation process (approximately four years), which is divided into 3 phases. The first feedback was obtained through a proof of concept, during which we applied the common concepts to express traditional, i.e. ISO/IEC 29110 Basic profile, and agile, i.e. Scrum, practices. The second feedback and improvements came after conducting a collaborative workshop held with 16 practitioners from software development organizations and academy. The purpose of the workshop was to evaluate pertinence, appropriateness and proficiency of the KUALI-BEH concepts through discussions, task solving and on-line surveys. During eight workshop sessions, the principal aspects of the framework were examined and applied in made-up situations. The feedback confirmed the appropriateness of the common concepts and led to improving terminology, adding new definitions and operations and changing operational rules.

The most substantial feedback and improvements were obtained during three case studies where KUALI-BEH was introduced into real-life projects and five OMG Task Force Meetings in the ESSENCE standardization context. During one of the case studies, KUALI-BEH was implemented to integrate and document a game-based method related to the Inception phase of software projects. The method was developed in a Mexican organization in response to the time- and money-consuming task of requirements specification and difficulties related to stakeholders' involvement in projects. They found that the inclusion of games and ludic materials helps the stakeholders as well as the team, the former define what they actually want, and the latter elicit all the necessary information from the stakeholders. However, this effort was not formalized nor was it possible to replicate it by other or new members of the organization. The value brought by KUALI-BEH consisted in converting it into a well-defined, congruent and replicable method (Morales-Trujillo, Oktaba, & González, 2014). Using the framework's terminology, the ALPHA Practice Authoring reached the Well-formed state, and, when the case study concluded, it reached the In-use and In-optimization states. This work was followed by a design and construction of a KUALI-BEH software tool (Medina-Díaz, Morales-Trujillo, & Oktaba, 2016).

Also, a formalization through description logics is proposed in (Morales-Trujillo et al., 2018). The objective pursued by the formalization was to foster communication, computational inference and reuse of practitioners' knowledge, which is achieved using domain ontologies as a means to represent and share common SE knowledge, Situational Method Engineering (SME) principles for defining method properties and operations, and Description Logics as a knowledge description language. The effectiveness of ontologies in addressing terminology concerns has been demonstrated in many fields (Guizzardi, Falbo, & Pereira Filho, 2002), therefore the formalization, based on an ontology, KB-O, collects harmonized definitions of the KUALI-BEH common concepts.

One of the latest applications of KUALI-BEH was related to defining practices and methods in a project at the Coordination of University Digital Collections (CCUD for its initials in Spanish) at the National Autonomous University of Mexico (Zavala-Correa, Morales-Trujillo, & Oktaba, 2016). The CCUD started to use KUALI-BEH after the standard was published, and no training or intervention from the researchers was requested.

The project's objective was to create a platform and a website portal to manage numerous University collections so they would be easily visualized and accessed by anyone. KUALI-BEH was introduced when the project had already started, and there was an urgent need to make explicit the way the people from the CCUD development and management areas carry out their activities. Apart from defining their methods without interrupting the usual way of working, the practitioners followed an ordered development process and generated customized documentation according to the framework. A recent informal chart with a practitioner from CCUD revealed that they adopted KUALI-BEH for other projects since they find particularly useful its flexibility and "non-imposing" approach.

In November 2014, the CCUD software engineering team identified the practices to be authored and started their expression using backtracking techniques and interviews with the rest of the CCUD members. The expressed practices reflected the real way of working, not hidden anymore, and were "In-use" during the next months. In August 2015, all the authored practices reached the "Agreed" state. Since then, the organizational method has been "In-optimization". During this period, the authored practices experienced addition of tasks and application of adaptation operations to certain practices. They created a global requirements specification method for wide contexts by fusing six practices related to requirements. Besides, they modularized the system architecture definition practice by splitting it into simpler and more specialized practices. The project involved 17 people and the defined practices and methods are still in use.

In April 2016, the method reached the "Consolidated" state. The following are the benefits of having developed a solid organizational way of working as reported by the members of CCUD: (i) better work distribution was achieved; (ii) the CCUD envisioned the whole project, which made them aware of its complexity and importance; (iii) the team's reports to the project manager and the CCUD directors were improved and simplified; (iv) training of new members joining the CCUD was reduced in time and effort by 40%.

More information on the case studies, the OMG standardization process and the post-standard experience can be found in (Morales-Trujillo, Oktaba, & González, 2015; Morales-Trujillo, Oktaba, & Piattini, 2015a, 2015b).

4.4 THREATS TO VALIDITY

This section discusses various threats to validity that were taken into account and counteracted during the development of KUALI-BEH. They concern the construct validity, internal and external validity and reliability.

On the one hand, the artifact's construct validity was ensured since KUALI-BEH was created following the mandatory requirements requested by the OMG in the FACESEM RFP; in addition, it was validated several times by the OMG Analysis and Design Task Force. On the other hand, multiple data sources were used in order to provide evidence and respond to the research objective. Those sources were interviews, direct observations, surveys and work artifacts, thus covering direct, indirect and third-degree methods defined by Runeson et al., 2012.

As for the internal validity, the results of the case studies proved that the objective for which KUALI-BEH was created was achieved, thus allowing us to accomplish our goals and the participants' needs. Several factors need to be mentioned. First, the participants of the workshop and case studies have a representative profile of practitioners working in VSEs. They possess different degrees of experience, going from juniors to seniors, as well as their age and educational background are diverse. These factors account for the trustworthiness of the collected surveys and the lack of bias towards KUALI-BEH. Although the number of participants is not large, the sample was representative.

The external validity was also addressed. In spite of the limited number of participant organizations, each of them is as a typical software developer entity and share the main characteristics of VSEs who are the target audience of KUALI-BEH. Moreover, the selection of participants was not intentional; the participant organizations themselves expressed their interest in taking part. Finally, thanks to the OMG and SEMAT, the obtained results were shared with researchers and practitioners from other countries.

The reliability was ensured by constantly triangulating the results to third parties, such as colleagues and members of the research group. The participants were also informed of the results and lessons learned, which were extensively disseminated in case study reports. Besides, we provide a detailed plan that guided the case studies activities for anyone who wishes to replicate them.

Returning to (Runeson et al., 2012), we found the following arguments to counteract the threats to this proposal's validity: (i) prolonged involvement, especially in second and third case studies (3 and 6 months respectively); (ii) involvement of assistants and different data sources; (iii) involvement of two researchers and peer debriefing all case studies; (iv) sharing of work products and documents with the participants of case studies; and (v) defining a version control strategy so the audit trail mechanism was easy to follow and successful.

5. CONCLUSIONS AND FUTURE WORK

Situating software engineering practices on universal elements "will give solidity and soundness to things built with them. Formally defining practices will identify the commonalities between them, leading to a better way to collect and share knowledge" (Jacobson, Ng, & Spence, 2007).

The knowledge collected in process reference models, standards and models implements a top-down approach that overlooks the way practitioners actually work and is not helpful for expressing their own practices. The contribution of KUALI-BEH consists in placing focus on practitioners' ways of working. Its bottom-up approach, contrary to renowned standards and models, supports practitioners in formalizing their tacit knowledge and building up a solid organizational *modus operandi*, which is expressed and stored as a collection of methods and practices. It promotes collaboration and communication among members of work teams, opens space for rethinking and improvement of practices, and minimizes the loss of knowledge. These benefits are particularly important for VSEs.

The Practice and Method Authoring ALPHAs represent a smooth transition from one state to another, totalizing six states. Starting with the Practice Authoring ALPHA and then moving to the Method Authoring ALPHA, practitioners authorize their tacit ways of working, making them explicit and tangible. The ALPHAs guide practitioners on the path of reaching a well-formed method of their own that will be suitable in any project of a particular organization and, at the same time, can be customized according to new needs.

After validating the bottom-up approach of KUALI-BEH we can conclude that it is a valuable alternative for practitioners and small organizations since it provides a first step to bridge the gap between software engineering theory and practice. KUALI-BEH permits small organizations to create an organizational method repository of their own knowledge, to foster their maturity and to gradually introduce them to the adoption of standards and reference models. On the other hand, adoption and usage of KUALI-BEH requires little effort from practitioners and no financial investment from the organization. Moreover, with the organizational way of working expressed, the VSE can establish inter-organizational collaborations through the execution of business processes between heterogeneous and autonomous organizations (Tello-Leal, Chiotti, & Villarreal, 2014).

In addition, a deeper reasoning about ways of working is possible, which provides solutions within reach for practitioners and helps organizations to effectively achieve their goals in the industrial context.

The potential impact of KUALI-BEH can be traced on two fronts: within academia and outside it. On the one hand, collecting and clearly defining the software project common concepts contributes to the body of theoretical knowledge of Software Engineering. On the other hand, improving VSE's knowledge management process and helping them to formalize their way of working through the

KUALI-BEH's bottom-up approach gives them a greater potential on the industry market. In fact, the organizations that participated in the case studies achieved tangible benefits. Since VSEs account for more than a half of all software developing organizations, the potential effect of adopting through bottom-up may have considerable positive consequences on the industry as a whole.

Finally, KUALI-BEH contributed actively to the SEMAT initiative and to the OMG specification process in the identification of the theoretical and practical universal elements (Morales-Trujillo, Oktaba, & Piattini, 2015c).

As future work, two lines are pursued: to develop a fully functional software that will automate the use of the framework and to continue spreading KUALI-BEH among organizations, especially VSEs. Both of them are meant to motivate practitioners to use KUALI-BEH and take advantage of their own expertise and knowledge.

ACKNOWLEDGMENTS

This work has been funded by the GEMA (Generation and Evaluation of Models for dAta Quality) supported by JCCM Consejería de Educación y Cultura y Deportes, y Fondos FEDER (SBPLY/17/180501/000293), the Graduate Science and Engineering Computing (UNAM), the grant scholarship program of CONACYT and the Postdoctoral Fellowships Program of DGAPA (UNAM).

CONFLICT OF INTEREST

The authors have no conflicts of interest to declare.

REFERENCES

Basri, S., & O'Connor, R. V. (2010). Understanding the perception of very small software companies towards the adoption of process standards. In *European Conference on Software Process Improvement* (pp. 153-164). Springer, Berlin, Heidelberg.

Basri, S., & O'Connor, R. V. (2011). Towards an understanding of software development process knowledge in very small companies. In *International Conference on Informatics Engineering and Information Science* (pp. 62-71). Springer, Berlin, Heidelberg.

Becker-Kornstaedt, U., Scott, L., & Zettel, J. (2000, June). Process engineering with Spearmint/sup TM//EPG. In *Software Engineering, 2000. Proceedings of the 2000 International Conference on* (p. 791). IEEE.

Brinkkemper, S. (1996). Method engineering: engineering of information systems development methods and tools. *Information and software technology*, *38*(4), 275-280.

Broy, M. (2011). Can practitioners neglect theory and theoreticians neglect practice?. *Computer*, (10), 19-24.

Clarke, P. M., Calafat, A. L. M., Ekert, D., Ekstrom, J. J., Gornostaja, T., Jovanovic, M., ... & O'Connor, A. (2016a). Refactoring software development process terminology through the use of ontology. In *European Conference on Software Process Improvement*, 47-57. Springer, Cham.

Clarke, P., Mesquida, A. L., Ekert, D., Ekstrom, J. J., Gornostaja, T., Jovanovic, M., ... & O'Connor, A. (2016b). An investigation of software development process terminology. In *International Conference on Software Process Improvement and Capability Determination*,351-361.Springer,Cham

Coleman, G., & O'Connor, R. (2007). Using grounded theory to understand software process improvement: A study of Irish software product companies. *Information and Software Technology*, *49*(6), 654-667.

Coleman, G., & O'Connor, R. (2008). Investigating software process in practice: A grounded theory perspective. *Journal of Systems and Software*, *81*(5), 772-784.

Dreyfus, H., & Dreyfus, S. (1986). Mind over machine: The power of human intuition and expertise in the era of the computer New York.

Edwards, J. S. (2000). Artificial intelligence and knowledge management: how much difference can it really make. *Proceedings of KMAC2000, Knowledge Management Beyond The Hype: Looking Towards The New Millennium, Operational Research Society, Aston University, Birmingham, UK*, 136-147.

Edwards, J. S. (2003). Managing software engineers and their knowledge. In *Managing software engineering knowledge*, 5-27. Springer, Berlin, Heidelberg.

Espinosa-Curiel, I. E., Rodríguez-Jacobo, J., & Fernández-Zepeda, J. A. (2016). Understanding SPI in small organizations: a study of Mexican software enterprises. *Journal of Software: Evolution and Process*, *28*(5), 372-390.

Franch, X., & Ribó, J. M. (1999, October). Using UML for modelling the static part of a software process. In *International Conference on the Unified Modeling Language* (pp. 292-307). Springer, Berlin, Heidelberg.

Guizzardi, G., Falbo, R. D. A., & Pereira Filho, J. G. (2002). Using objects and Patterns to implement domain ontologies. *Journal of the Brazilian Computer Society*, *8*(1),43-56.

Harmsen, A. F., Brinkkemper, J. N., & Oei, J. H. (1994). *Situational method engineering for information system project approaches* (pp. 169-194). University of Twente, Department of Computer Science.

Henderson-Sellers, B., Gonzalez-Perez, C., Mcbride, T., & Low, G. (2014). An ontology for ISO software engineering standards: 1) Creating the infrastructure. *Computer Standards & Interfaces*, *36*(3), 563-576.

Knowledge Based Systems, Inc., D. (1993). IDEF0: *Integration definition for function modeling*.

International Organization for Standardization. (2004). ISO/IEC 15504: Information technology – Process assessment.

International Organization for Standardization. (2005). ISO 9000: Quality management systems – Fundamentals and vocabulary.

International Organization for Standardization. (2007). ISO/IEC 24744: Software Engineering – Metamodel for Development Methodologies.

International Organization for Standardization. (2008a). ISO/IEC 12207: Systems and software engineering – Software life cycle processes.

International Organization for Standardization. (2008b). ISO/IEC 15288: Systems and software engineering – System life cycle processes.

International Organization for Standardization. (2010). ISO/IEC TR 24774: Systems and software engineering – Life cycle management – Guidelines for process description.

International Organization for Standardization. (2012). ISO/IEC 29110-5-1-2: Software engineering – Lifecycle profiles for Very Small Entities (VSEs) – Management and Engineering Guide: Generic profile group: Basic Profile.

Jacobson, I., Ng, P. W., & Spence, I. (2007). Enough of Processes-Lets do Practices. *Journal of Object Technology*, *6*(6), 41-66.

Jacobson, I., Ng, P. W., McMahon, P. E., Spence, I., & Lidman, S. (2012). The essence of software engineering: the SEMAT kernel. *Communications of the ACM*, *55*(12), 42-49.

Johnson, P., Ekstedt, M., & Jacobson, I. (2012).Where's the theory for software engineering? *IEEE software*, *29* (5), 96-96.

Krdžavac, N., Gašević, D., & Devedžić, V. (2009). Model driven engineering of a tableau algorithm for description logics. *Computer Science and Information Systems*, *6* (1), 23-43.

Laporte, C. Y., Alexandre, S., & O'Connor, R. V. (2008). A software engineering lifecycle standard for very small enterprises. In *European Conference on Software Process Improvement* (pp. 129-141). Springer, Berlin, Heidelberg.

Lee, J., Gruninger, M., Jin, Y., Malone, T., Tate, A., Yost, G., & PIF Working Group. (1998). The process interchange format and framework. *The knowledge engineering review*, *13*(1), 91-120.

Medina-Díaz, R., Morales-Trujillo, M. & Oktaba, H. (2016). Computer Schema for Semi-automated Verification of Methods and Practices. EUSICS'16 in *International* Symposium on Intelligent Computing Systems. Mérida, Yucatán, México, pp. 103-109.

Morales-Trujillo, M. E., Oktaba, H., & González, J. C. (2014). Improving software projects inception phase using games ActiveAction workshop. In 2014 International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE). (pp. 1-8). IEEE.

Morales-Trujillo, M. E., Oktaba, H., & González, J. C. (2015). Taking Seriously Software Projects Inception Through Games. In International Conference on Evaluation of Novel Approaches to Software Engineering, 109-124.

Morales-Trujillo, M., Oktaba, H., & Piattini, M. (2015a). Using technical-action-research to validate a framework for authoring software engineering methods. In Proceedings of the 17th International Conference on Enterprise Information Systems, 2, pp. 15-27.

Morales-Trujillo, M., Oktaba, H., & Piattini, M. (2015b). Validating a Software Engineering Framework Through Technical-Action-Research in Union with Case Studies. In International Conference on Enterprise Information Systems, pp. 303-327. Springer.

Morales-Trujillo, M. E., Oktaba, H., & Piattini, M. (2015c). The making of an OMG standard. Computer Standards & Interfaces, 42, 84-94.

Morales-Trujillo, M., Oktaba, H., Hernández-Quiroz, F., & Escalante-Ramírez, B. (2018). Towards a Formalization of a Framework to Express and Reason about Software Engineering MethodS. Computing & Informatics,37(1),109-141.

NYCE. (2011). NMX-I-059-NYCE-2011: Modelo de Procesos para la Industria del Software (MoProSoft).

Object Management Group. OMG. (2008). Software and Systems Process Engineering Meta-Model. Version 2.0, formal/2008-04-01, Needham, MA, USA. Retrieved September 2017, from: http://www.omg.org/

Object Management Group. (2011). A Foundation for the Agile Creation and Enactment of Software Engineering Methods RFP. Technical report. Needham, MA, USA.

Object Management Group. (2012). KUALI-BEH Software Project Common Concepts. Technical draft. Needham, MA, USA.

Object Management Group. (2014). ESSENCE – Kernel and Language for Software Engineering Methods. Needham, MA, USA.

O'Connor, R. V., & Basri, S. (2014). Understanding the role of knowledge management in software development: a case study in very small companies. International Journal of Systems and Service-Oriented Engineering (IJSSOE), 4(1), 39-52.

Oktaba, H., Piattini, M. (2008). Software Process Improvement for Small and Medium Enterprises: Techniques and Case Studies, 1-376.

Oktaba, H., García, F., Piattini, M., Ruiz, F., Pino, F. J., & Alquicira, C. (2007). Software process improvement: The COMPETISOFT project. *Computer*, *40*(10), 21-28.

Object Management Group. OMG. (2008). Software & systems process engineering meta-model specification. *OMG Std., Rev*, *2*, 18-71. Retrieved September 2017, from: http://www.omg.org/

Osterweil, L. (1987). Software processes are software too. In *Proceedings of the 9th international conference on Software Engineering* (pp.2-13). IEEE Computer Society Press.

Pease, A., & Carrico, T. M. (1997). Object Model Working Group (OMEG) Core Plan Representation–Request for Comment, version 2. *Defense Advanced Research Projects Agency.*

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

Pino, F. J., Pardo, C., García, F., & Piattini, M. (2010). Assessment methodology for software process improvement in small organizations. *Information and Software Technology*, *52*(10), 1044-1061.

Project Management Institute. (2008). A Guide to the Project Management Body of Knowledge: PMBOK Guide. *Global Standart*, Fourth Edition.

Radice, R. A., Harding, J. T., Munnis, P. E., & Phillips, R. W. (1985). A programming process study. *IBM Systems Journal*, *24*(2), 91-101.

Rout, T. (1999). Consistency and conflict in terminology in software engineering standards. In *isess* (p. 67). IEEE.

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

Schlenoff, C., Schlenoff, C., & Ray, S. (1996). Unified process specification language: Requirements for modeling process. US Department of Commerce, National Institute of Standards and Technology.

Schwaber, K. & Sutherland, J. (2017). The Scrum Guide – The Definitive Guide to Scrum: The Rules of the Game.

Software Engineering Institute. (2010). CMMI: Capability Maturity Model Integration, Software Engineering Institute.

Staples, M., Niazi, M., Jeffery, R., Abrahams, A., Byatt, P., & Murphy, R. (2007). An exploratory study of why organizations do not adopt CMMI. Journal of systems and software, 80(6), 883-895.

Tate, A. (1998). Roots of SPAR—shared planning and activity representation. The Knowledge Engineering Review, 13(1), 121-128.

Tello-Leal, E., Chiotti, O., & Villarreal, P. D. (2014). Software agent architecture for managing inter-organizational collaborations. Journal of applied research and technology, 12(3), 514-526.

Wang, S., Wang, W., Zhuang, Y., & Fei, X. (2015). An ontology evolution method based on folksonomy. Journal of applied research and technology, 13(2), 177-187.

Wang, Y. (2007). Software engineering foundations: A software science perspective. Auerbach Publications.

Wieringa, R. (2014). Empirical research methods for technology validation: Scaling up to practice. Journal of systems and software, 95, 19-31.

Wieringa, R., & Moralı, A. (2012). Technical action research as a validation method in information systems design science. In International Conference on Design Science Research in Information Systems. Springer, Berlin, Heidelberg,220-238.

Zavala-Correa, K., Morales-Trujillo, M. & Oktaba, H. (2016). Customized Methodology for Development of an Open Data Portal. International Conference on Software Engineering Research and Innovation, pp. 67-75.