Computer Science and Information Systems 13(3):827–847 DOI: 10.2298/CSIS160801038G

MyPMP: A Plug-in for Implementing the Metamodeling Approach for Project Management in Small-sized Software Enterprises

Ivan Garcia[1], Carla Pacheco[1], Magdalena Arcilla-Cobián[2], and Jose A. Calvo-Manzano[3]

[1] Division de Estudios de Posgrado, Universidad Tecnológica de la Mixteca, Carretera a Acatlima Km. 2.5, 69000. Oaxaca, Mexico.
{ivan, leninca}@mixteco.utm.mx
[2] Escuela Técnica Superior de Ingeniería Informática, Universidad Nacional de Educación a Distancia, Ciudad Universitaria, 28040. Madrid, España.
marcilla@issi.uned.es
[3] Escuela Técnica Superior de Ingenieros Informaticos, Universidad Politecnica de Madrid, Campus Montegancedo, 28660. Madrid, España.
joseantonio.calvomanzano@upm.es

Abstract. Nowadays, with the recurrent demands of high quality, delivery on time and no extra costs, the task of managing a software project could be extremely complex for any software enterprise. Furthermore, small-sized software enterprises face several problems (e.g., lack of knowledge, human and financial resources, time, and size of staff) that, undoubtedly, make this task more difficult. In this context, obtaining a simplified version of the management activities can be a helpful alternative for these enterprises. In this way, that an inexperienced project manager can define the management process that best fits with a particular project is not an easy task. Thus, this paper introduces the metamodeling approach in order to help project managers to define a process for managing a software project. Therefore, with the aim to validate its feasibility an add-in program was developed as a part of a case study. The achieved results show an important reduction in project's effort and time needed to develop a new software product.

Keywords: project management, metamodeling approach, plug-in, small-sized software enterprises.

1. Introduction

In the context of the software industry, the quality of a product depends on the process used to complete the project. Therefore, there is always a lot of pressure on software projects to become more productive and efficient. Nevertheless, software projects have been characterized by facing many issues and problems throughout their life cycles. According to [1], managing a software project can be extremely complex, and its success frequently relies on many personal, team, and organizational resources. In this regard, research by Mir and Pinnington [2] argues that project success has been conceptualized for many studies as a uni-dimensional construct related with meeting

budget, time, and quality whereas other studies have considered project success a complex and multi-dimensional concept encompassing many more attributes. However, in spite of the advancement in the definition of project management processes, tools, and computer systems, the rate of software projects success has not significantly increased. Moreover, a recent study stated that the value of project management depends on the country, culture, industry type, size of organization, and organizations' needs [3]. In fact, nowadays we still have not figured out how to substantially improve the software project success rate. For example, Cerpa, Bardeen, Astudillo, and Verner [4], have identified some factors influencing software project outcome: organizational structure; communication with customer/users; scheduling and project budget; customer satisfaction; product quality; leadership; software development methodologies; and the project management process and tracking tools, among others. Many of these factors are obviously related to project management. Thus, projects are the cornerstone of all commercial activities in small-sized software enterprises.[1] Therefore, these organizations must conclude many software projects to achieve their business and financial goals using a carefully planned and controlled process. In this regard, the nature of project management required by small-sized software enterprises is very different than the traditional forms suggested for larger enterprises. This is, small-sized software enterprises require less bureaucratic forms of project management than those used by larger, traditional organizations [5]. Unlike large enterprises, small-sized software enterprises do not have enough staff to develop functional specialties that would enable them to perform complex and secondary tasks to improve the software projects management and, as consequence, the products quality. The employees of these enterprises perform multi-tasks, so it is common that software projects are managed by people for whom project management is not their main area of expertise. Furthermore, due to the necessity of being competitive, small-sized software enterprises undertake many projects that are often managed by amateurs that are leading them, as consequence, to the failure. In this scenario, executing a proper process for software project management is an important challenge. As O'Connor and Laporte say: *"a good project management cannot guarantee project success, however a bad project management usually results in project failure"* [6], therefore good process and proper techniques can improve the project chances of success. In this sense, it is common that project management is seen as one of the key strategies for managing the success of projects and organizations by recognizing the value of project management approaches and the necessity of skilled employees for executing the projects.

Moreover, the work of the project is always carried out by planning, executing, and monitoring and controlling processes [7]. In fact, many researchers agree that an effective project management involves repeated performance of these processes. But, how can a small-sized software enterprise, without knowledge and experience, define a repeatable process for managing its projects? Research by Turner, Ledwith, and Kelly

[1] A small-sized software enterprise —independently financed and organized companies with fewer than 50 employees— is a privately owned and operated business that typically has a small number of employees. In most countries around the world, the legal definition of a small enterprise is determined by the government, which sets the criteria to be used by the national market in making small business determinations [9].

[8] has shown that different versions of project management might be required for small-sized enterprises (a micro-lite version). It means, a simplified version of project management practices for managing the work, duration, and used resources in its core. Additionally, project management procedures used by small-sized software enterprises will also include status reports for cost and time, work breakdown structure, and task sharing. In this context, it is true that some project management procedures for small-sized enterprises have been developed and these have paid attention in the human aspects of project management (e.g., team working, unskilled staff, motivational issues). Nevertheless, these are also too bureaucratic and complex for small-sized enterprises. On the other hand, traditional methodologies usually provide guidance on what steps to follow in order to manage a project for obtaining a desired software product. Even so, an extra effort is required when trying to match a given methodology with the necessities of a small-sized software enterprise. In this scenario, metamodels have been proposed as an alternative tool for proper understanding of the methodologies/processes of Software Engineering through modeling. In essence, using metamodels means modeling a methodology as if it were any other system, applying the same modeling ideas and procedures that are usually applied to business applications or other software-intensive systems [10]. Therefore, a metamodel can provide a conceptual model that leads to a formalization of concepts, in order to provide formal definitions of a process that can be used to generate tools for supporting the software development process [11]. In this regard, this paper introduces a metamodel that enables project managers to define a simplified version of the basic project management activities in the context of small-sized software enterprises. The rest of this paper is organized as follows: Section 2 provides a detailed description of metamodels within the Software Engineering context and outlines the related work related to the use of metamodels for project management. In Section 3 our metamodel is presented in order to provide an easy description for project management in the context of small-sized software enterprises. With this aim in mind, we make our approach more practical by providing an add-in program for Microsoft Project® 2007. The detailed information is also provided. Furthermore, in order to illustrate the feasibility of our approach in small-sized software enterprises, quantitative results of a case study are also provided in Section 4. Finally, Section 5 draws the conclusions and main findings of this study.

2. Related work on metamodeling approaches in project management

Over the last decade we have witnessed the creation of a more rigorous support to the various areas of software project management. For example, some of these areas are the introduction of artificial intelligence for project scheduling [12, 13], the conception of agile management [14, 15], the reuse of software project manager's experience [16, 17], and the modeling as support for software development [18, 19]. In the latter case, Henderson-Sellers [10] affirms that a model is an abstraction that represents some view of reality, necessarily omitting details, for some specific purpose and which may be used to document existing situations or to describe situations that might occur. Therefore, some researchers have created descriptive models for depicting and documenting the

software project management activities in order to get a finer granularity and to correctly understand the state of the practice [20]. With this idea in mind, the benefit for small-sized software enterprises from using these models is to get a better understanding of the project management activities by obtaining a "simplified" version of the current practice. In this regard, a metamodel can be used too to increase the general understanding of the project management process. Thus, a metamodel is also a model and it is generally defined as a "model of models" or, equally, "a model of a set of models" [10]. In this context, the use of metamodels has been increasingly explored by diverse researchers in the context of Software Engineering. Nevertheless, in spite that diverse areas of Software Engineering have been addressed with the metamodel approach (e.g., requirements engineering [21–23], software process assessment [24, 25] and measurement [26, 27], software process improvement [28–30]), the software project management has been poorly analyzed. For example, research by [31] introduced the PROMONT ontology for project management in order to build a common understanding of project related terms and methods, and thus, facilitating the management of projects conducted in dynamic virtual environments. In this regard, according to Henderson-Sellers the domain ontologies can be used to create a vocabulary for a specific application domain (e.g., project management) ensuring that elements in the model have well-defined semantics. Otherwise, meta-ontologies or foundational ontologies, which are equivalent to a metamodel, encapsulate the concepts needed for creating domain ontologies [10]. Thereby, PROMONT provided means for expressively stating axioms and specifications of the concepts and relations in project management.

Moreover, in [32] a metamodel for agile project management is proposed within an educational context. Concretely, this study defines a workflow for teaching and practicing the agile project management methods by using the LEGO® bricks building concept. Thus, bricks are used as a medium to transmit agile principles practices to participants with various background knowledge and experience. The proposed workflow was used for designing an educational workshop and it is the result of a metamodel of methods —agile project management and bricks building. Therefore, this metamodel conceptualizes the analogies among software development, software engineering principles, bricks building process, and agile project management.

Similarly, research by Callegari and Bastos [11] presents a metamodel for software project management based on the PMBOK® Guide and its integration with RUP (Rational Unified Process). Although the reference documents for both models describe different kinds of models or metamodels, the purpose of this study was to provide a complete understanding of the core concepts of both models by providing a mapping between them, and by proposing an integrated model that can guide practitioners towards a better software project management and, as consequence, to obtain high quality products. In this regard, the metamodel for project management provides concepts that cover human and physical resources, activities, deliverables, as well as time and organizational concepts and their associated classes.

Finally, research by Thiemich and Puhlmann [33] combines the BPM (Business Process Management) methodology and Scrum (an agile methodology for project management) to introduce a metamodel for agile BPM projects. This metamodel consists of three core aspects: project approach, artifacts, and methods, and it indicates how they are connected with the management process in order to enhance the reflected adaption of agile principles in BPM projects. As a result of this research, the first two

steps of a traditional BPM lifecycle (i.e., modeling and implementation) are merged together, resulting in "better" processes that can be implemented according the organization's necessities (i.e., the processes that are really needed). Since the organization gets a better understanding of what they really want in each iteration, the fuzziness of the to-be processes is cleared up in early stages.

In this regard, it is important to highlight that in the previous studies, improving project management within enterprises is assumed to be made through the creation of metamodels which include specific project management practices, such as work breakdown structures or earned value management, as well as simplification of activities that would help to improve the understanding of management activities, including the description of project management processes, tools, and techniques, or the designation of formal titles and roles for those in charge of projects, and their adequate training. With this aim in mind, we have developed a metamodel to define a "lite" version of the project management process for the context of small-sized software enterprises. Moreover, we have implemented our approach into an add-in program for Microsoft Project® 2007 in order to simplify the management process and to manage the knowledge generated during the software development. Additionally, it is important to mention that this approach does not require a large investment of cash capital to establish within small-sized software enterprises.

3. Developing MyPMP for small contexts

As we said before, the advances of Software Engineering have inspired the use of modeling techniques to different areas; in fact, various researchers have seen this as a chance to represent the knowledge of particular issues of their research (e.g., risk identification, requirements analysis, project management) in the form of metamodels. In this regard, the benefits of a metamodel may include: domain concepts are easier to apply for novices or young unexperienced employees (i.e., concepts would be presented in the metamodel instead of having to look for them in a dispersed collection of books, papers, models); increased portability of practices across supportive management tools; and better communication among project manager and employees [34]. Furthermore, according to Othman and Beydoun [35] a metamodel is a fundamental building block that makes statements about the possible structure of models. Moreover, a metamodel is usually defined as a set of constructs of a modelling language and their relationships, as well as constraints and modelling rules without necessarily the concrete syntax of the language [34].

3.1. The proposed metamodel

In our context, and according to Henderson-Sellers [10], metamodeling in current Software Engineering follows one of two possible architectures. The OMG (Object Management Group) architecture that is based on strict metamodeling [36] wherein the only relationship between levels is called "instance of" (left side of Fig. 1). In OMG standards, for example, an M0 object is said to be an instance of a class in level M1; a

class in level M1 is said to be an instance of a metaclass in level M2 and so on. On the other hand, the right side of Fig. 1 shows an alternative multi-level architecture that introduces the *powertype* pattern as used in ISO/IEC 24744 [37]. Thus, an object facet provides attributes for Method Domain entities while the class facet provides specification of attributes that are given a value in the Endeavour Domain. Furthermore, the use of *powertypes* permits both instance-of and generalization relationships between levels. Thus, in the context of defining methodologies for software development, this pattern combines the main advantages of other metamodeling approaches and enables the integration of documental aspects into the methodology [38].

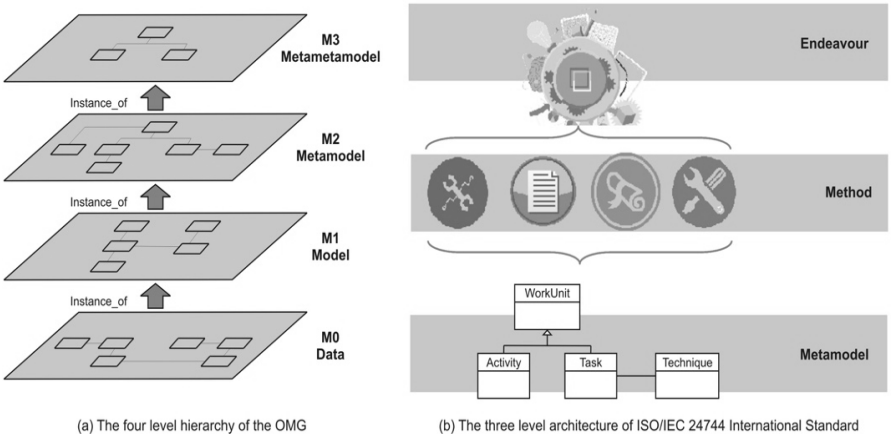

Fig. 1. Architectures for metamodeling in the context of Software Engineering [5]

In this regard, and taking into account the ISO/IEC 24744 recommendations, we have proposed a conceptual architecture to build our metamodel. This architecture was constructed by a combination of bottom-up and top-down analysis and best practice. The representation of Fig. 2 aims to integrate all the process-related elements through four layers and leads the formalization of concepts in the context of small-sized software enterprises. Therefore, the aim of this conceptual architecture is to provide a mechanism for improving the understanding and implementation of a management process through four layers of abstraction. These four layers of abstraction are defined as follows:

– Level M0: The data generated by the projects represent the "real word" in the architecture. Thus, all historical data obtained by developing successful or failed projects are useful to learn how to manage them and, in a long-term, to predict the process performance.

– Level M1: An adjustment of complexity of practices recommended by process reference models (e.g., the PMBOK® Guide) is needed to provide to small enterprises a simplification of the practice. In this sense, our metamodel provides a set of scripts, templates and guidelines (called project planning assets) to support the project manager's work. Additionally, the process evaluation is a crucial activity for this level, because it is not possible to define (or adapt) a new process without

determining the current state of the practice (i.e., weaknesses and strengths of project management within enterprise).

– Level M2: The definition of a new process (or the adaption of an existing one) is related to the necessity to adequate a simplified description of this in a specific context. The pattern concept explored by González-Pérez and Henderson-Sellers [38] is introduced in M2 for enabling project managers to take, from the concepts defined in M1, those useful to develop a specific project. This level defines a Process Asset Library (PAL) to support project management good practices within the participant small-sized software enterprises and to enable project managers the definition of an enacted process.

– Level M3: The repository of all previous meta-elements uses instances at different levels, as an analogy to the SPEM metamodel [39], to create different versions of a process from other processes that may be previously defined.

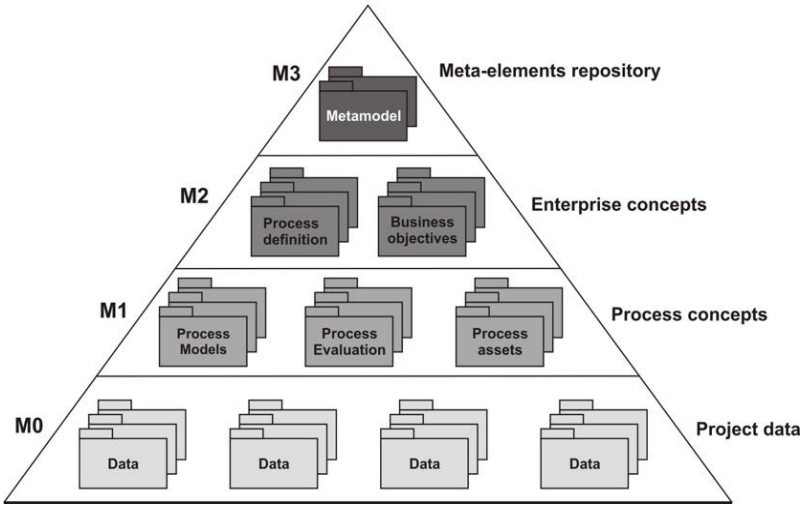

Fig. 2. A conceptual architecture for our metamodel

This conceptual architecture establishes the necessity of establish a set of common concepts to be used in the metamodel. In this sense, the PMBOK® Guide joins the knowledge of proven traditional and widely applied practices on project management, and it has been promoted by the Project Management Institute, a non-profit professional association, which primary goal is to advance the practice, science, and profession of project management [40]. Thus, in the context of introducing the project management process in small-sized software enterprises, in preparation for eventually managing larger and complex projects, the PMBOK® Guide has been widely explored. Therefore, we have focused our efforts in developing a metamodel for the *Project Scope Management*, *Project Time Management*, and *Project Cost Management* from the ten knowledge areas of PMBOK®. It is important to mention that by definition the PMBOK® Guide was not created to be a process and, as consequence, does not specify how to perform the activities on a software project for the development of a high quality

product and, hence, achieve the project success. Taking into account this situation, our metamodel was created with the aim of facilitating the definition of project management in the context of small-sized software enterprises. Moreover, this metamodel summarizes the essential elements of project management that a small-sized enterprise can use to begin to manage their projects. Furthermore, the main idea of this research is that with this metamodel a small-sized software enterprise can begin to mature at project-level and, with its recurrent use in the future, mature at process-level. With this aim in mind, we initially pretend to model all the basic process elements of the PMBOK® Guide, for managerial activities, and MoProSoft® [41], for productive activities. Consequently, the metamodel depicted in Fig. 3 was built using UML class diagrams.

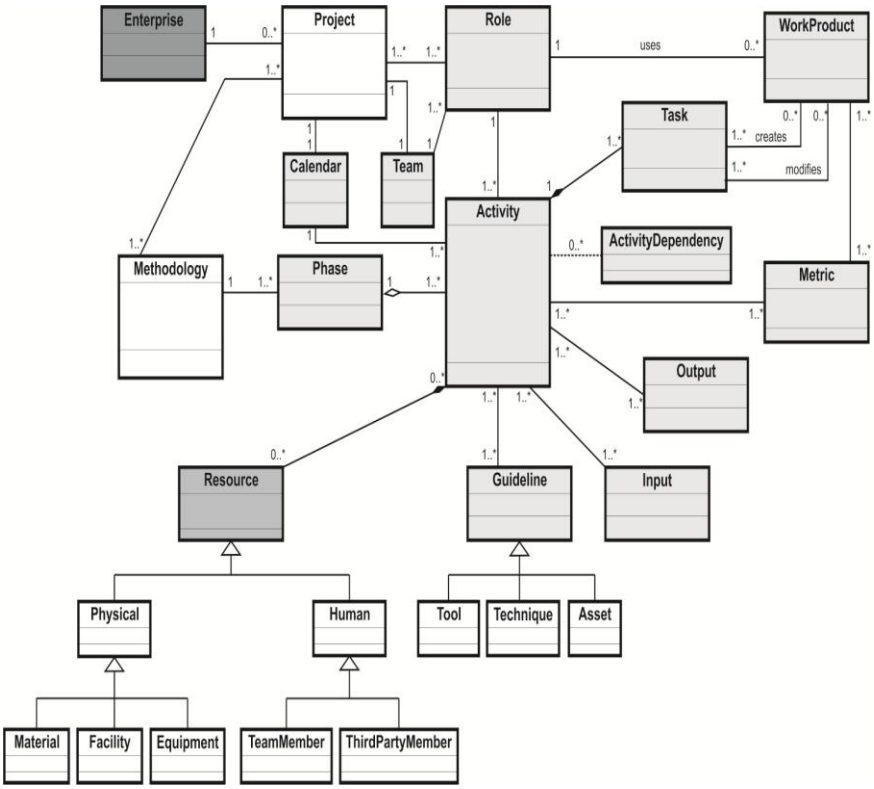

Fig. 3. The metamodel approach for project management in small-sized software enterprises

To better explain the metamodel, our description begins from "basic" concepts of management depicted in the PMBOK® Guide and their relationships to define a simplified management schema. In this regard, through this metamodel, a ***small enterprise*** can define a process to develop a ***project*** by integrating small teams that perform activities by following, or trying to follow, at least, a ***methodology*** (e.g., MoProSoft®). As we said before, it is common that a software development methodology states "what to do", but not "how", neither "who does it". Therefore, these

methodologies are composed of a collection of ***phases***. Each phase is composed by a set of ***activities***. It is noteworthy that the architecture of our metamodel does not establish by default a set of strict steps or activities for project managers. According to the conceptual architecture shown in Fig. 2, when a request for a specific project is created, it is necessary that the project manager defines (i.e., instantiates) all the elements of his own process. This process should take into the account the necessities and complexity of that project. At the same time, activities can be broken down into more specific actions called ***tasks*** that may require ***work products*** to facilitate their execution. Activities may have ***dependencies*** between them, which help to define the order in which they should be executed within the project. Additionally, the activities are usually supported by some kind of ***guideline*** (e.g., a tool, technique, artifact, process asset) so that it is possible to verify the procedure to be performed before its execution. The activities typically produce ***outputs*** (e.g., deliverables, documents, artifacts), and depend on ***inputs*** to generate a result.

Moreover, the ***roles*** in the metamodel are divided according to the type of activity to be performed (i.e., a managerial activity or a productive activity). Thus, each activity must be performed by one or more roles, something that is typical within small teams. A similar relationship can occur with the ***resources*** (physical or human) because an activity may depend on certain infrastructure and knowledge to achieve the project's objective. Furthermore, any given activity has only one responsible (as it is indicated by the standards and methodologies for project management), thus we are trying to eliminate the common problem in small-sized software enterprises when roles are constantly changed for each project. An activity can use a work product as input or output, but it can also modify an existing one; thus, this concept has been extended to allow three distinct associations in the metamodel: *generates*, *modifies*, and *uses* as inputs (e.g., an activity can generate a work product, or modify or use an existing one).

It is important to mention that it has also been indicated the association "modifies" between the work product and the role, because it is the employee who plays that role the one who has the possibility to change an artifact without performing any work. This consideration was also added to enable automation of an instantiated process considering the fact that it is not possible to eliminate an activity that creates a work product if there is another activity that modifies or uses as input the same product. Each work product must display information about its version (i.e., configuration management) and its type: "external" (when it is subjected to the approval of a person or client) or "internal" (when is total or partially delivered to the role that is responsible of that activity). Finally, a work product and an activity should use ***metrics*** to be verified and to determine its successful use, respectively.

In this regard, this metamodel provides a "lite" version of project management highlighting three basic concepts:

- Development activities, for helping employees to perform their work, including the sequencing of those activities and the interfaces between them. The employees of the small-sized software enterprises learn to explore and understand that a development activity is an artifact (e.g., a requirements document, a formal specification, program code, and test cases), and the recurrently use of this forces each employee to gain experience from scratch.

– Management activities, for helping project managers to plan and control all the development activities from the earliest conception stage to the final support stage of the project lifecycle.
– Tools and techniques, for supporting the activities execution (both managerial and productive) by defining assets, information, and resources.

From the inputs and outputs products proposed by MoProSoft®, and from the inputs, techniques, and outputs proposed by the selected processes of the PMBOK® Guide, we have designed a Process Asset Library (PAL) to support the performance of our project management metamodel (see Fig. 4). Moreover, in the experience of different organizations, the definition and implementation of a well-structured and organized PAL is the key that enables organizations to have a culture focused on the maturity of their processes.

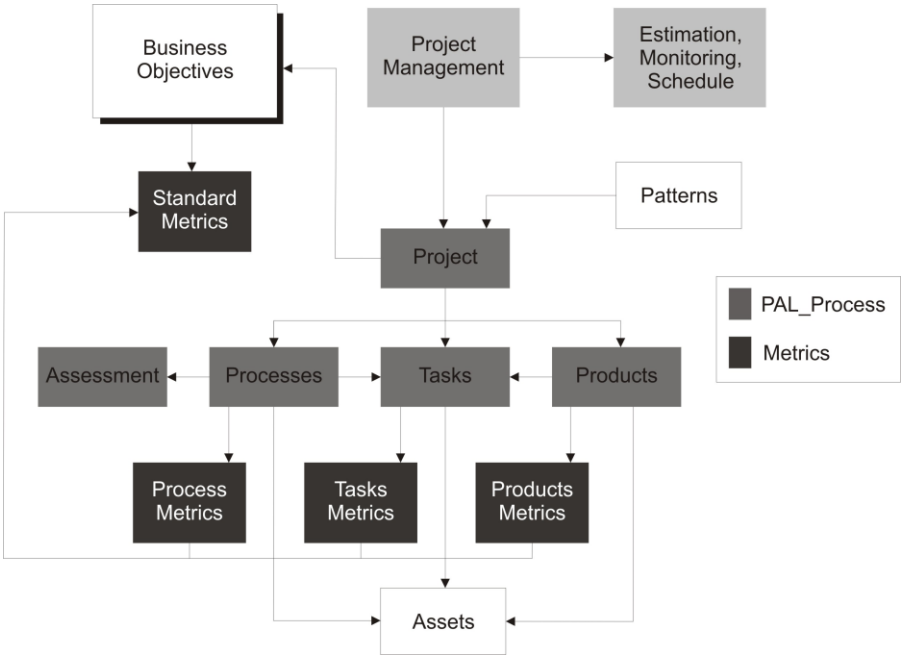

Fig. 4. Process asset library for supporting the project management metamodel

Therefore, following the metamodel from Fig. 3, a project management process from the PAL is defined in terms of tasks, products, metrics (of process, product, and task) and assets (of process, product, and metrics). Thus, all the assets are artifacts or mechanisms that provide support to perform the management process, products, tasks, and metrics. In this way, a project management process can be used by the projects of the small-sized software enterprise taking into account the guidelines and tailoring criteria defined in the levels of abstraction M1 and M2 of the metamodel.

3.2. Creating an add-in program for implementing the metamodel

The presented metamodel provides a conceptual architecture that enables project managers to define a unique process to assist them in project planning and control taking into account the concepts arising from the PMBOK® Guide and MoProSoft®. In order to demonstrate the feasibility of the proposed metamodel, we have developed MyPMP (My Project Management Process) as an add-in program for Microsoft Project® 2007. Based on this idea, the concepts coming from the metamodel were added to this prototype (which has already included as start point the simplifications of the PMBOK® Guide and MoProSoft® for managerial and productive activities, respectively). This choice allows small-sized software enterprises to take advantage of the features that are already implemented in Microsoft Project® with the proposed definition of a process for project management. In this regard, Microsoft Project® has been widely used by researchers and professionals for project planning and control, estimation, and analysis. While custom projects can be easily built with the current functionalities of Microsoft Project®, sophisticated and highly customizable macros can also be compiled using Visual Basic for Applications (VBA).

Once the add-in program has been installed, a pull-down MyPMP menu appears in the menu bar when Microsoft Project® is launched. As shown in Fig. 5, a project manager may select an option of interest from the menu and start a project taking into account the simplification of MoProSoft® activities to establish a software development lifecycle and the PMBOK® Guide to manage the project, such as the metamodel has defined it. The MyPMP program provides many customizable options for supporting the definition of a 'lite' version of a project management process, including selection of activities, initial planning values, pre-uploaded templates, and guidelines for a correct implementation in the context of small-sized software enterprises. In addition, all the data related to MoProSoft® and the PMBOK® Guide has been uploaded into a database to help project managers to begin from the scratch. This feature is provided as a guide for unskilled project managers to help them to arrange their projects into a suitable form for developing a high quality product. In this context, Fig. 5 shows five options to implement the proposed metamodel in a pull-down menu and one exit function has been developed for project managers who want to use Microsoft Project® without the add-in program.

The MyPMP functionalities can be used to define a new process, adapted to the enterprise and project contexts, for managing the software projects; to quantitatively analyze the projects and process performances, according to the defined metrics; and to print the created assets in case that the project manager wants to study or analyze a template, or read a recommendation for the proper execution of an activity. In this case, for example, in Fig. 5 the project manager has selected the option "Library of assets" to download an asset from four available categories: process, product, knowledge, or tools. All the assets are in PDF format to support the execution of an activity, the use of a tool, the understanding of a metric, or just for providing more specific knowledge about the software development and project management processes. For example, many small-sized software enterprises do not understand how to represent the project tasks as a work breakdown structure (WBS). In this regard, the M1 level of our metamodel provides the assets component to make each team member clear of the duties, powers, responsibilities and interests in the project. Thus, if the project has a good WBS, then as long as each

member complete his/her own part of work, the entire project can be completed [42]. In this context, MyPMP provides assets to explain, for example, the WBS importance and a template to correctly perform the task related to this tool.

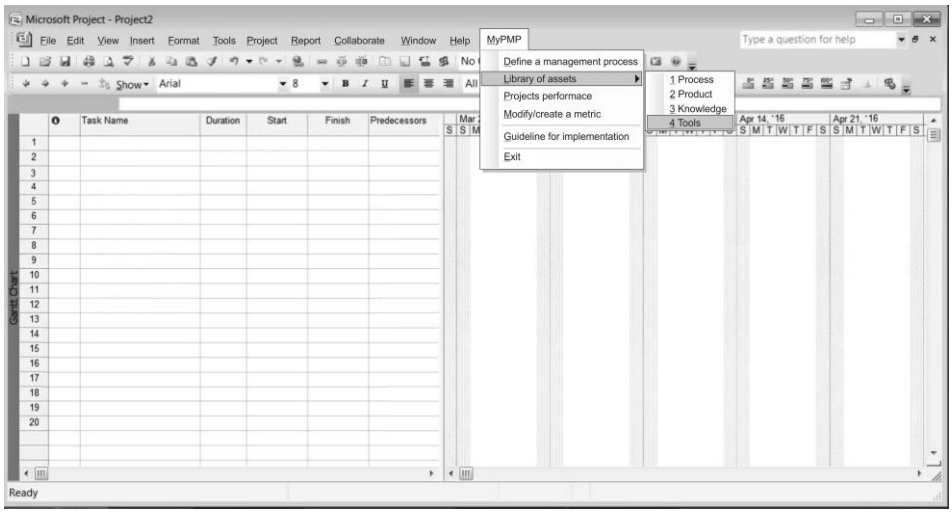

Fig. 5. MyPMP menu in Microsoft Project® environment

To satisfy the metamodel requirements, MyPMP uses a database with all the information of MoProSoft® and the PMBOK® Guide classified according to the levels defined by the layers of abstraction of the conceptual architecture. The MyPMP features have been purposely implemented to fit with the knowledge necessities of the small-sized software enterprises, otherwise project managers would have not a guidance to properly start, plan, and control a project, jeopardizing the project success. Thus, Fig. 6 shows how the add-in program enables project managers to define a process for project management taking into account the conceptual architecture of the metamodel. In this regard, once the project manager has selected to use the phase and activities of MoProSoft® and the PMBOK® Guide, he/she can delete anything that does not fit with the project and customize his/her own process (see Fig. 6 (a)). Additionally, if the project manager has selected the support option, MyPMP provides some advice about scheduling (see Fig. 6 (b)) according to some parameters previously configured (e.g., complexity, total time, size of team, cost, and more).

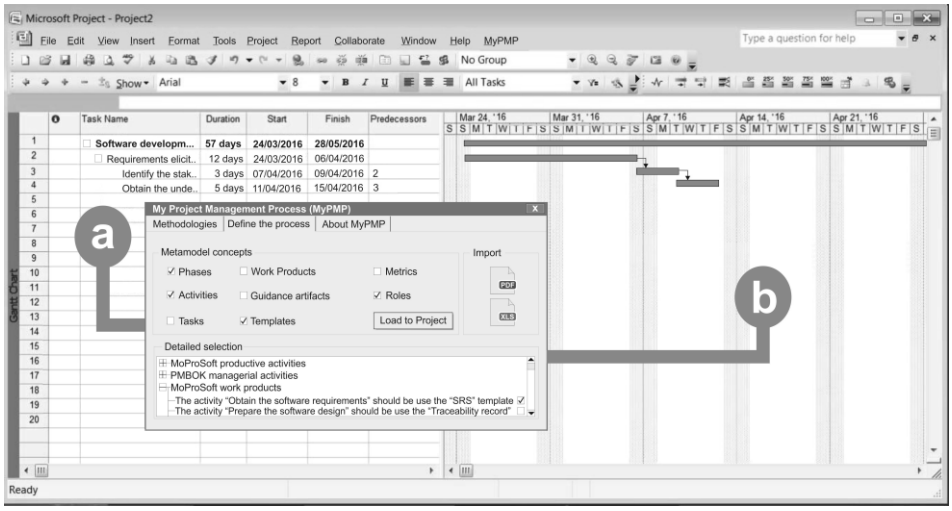

Fig. 6. Using MyPMP to define a project management process in Microsoft Project®

4. Case study

The aim of this case study was to evaluate, in an industrial context, how well MyPMP works when the metamodeling approach is implemented for project management within a small-sized software enterprise. In this regard, according to Kitchenham, Pickard, and Pfleeger [43], experimentation in Software Engineering increases the understanding of software quality characteristics and how to properly do it. Therefore, a case study was chosen as the experimental empirical strategy to be followed in the validation of the proposed metamodel through the use of MyPMP. Furthermore, in accordance with the recommendations of Wohlin, Höst and Henningsson [44], there are three different strategies to develop a case study:

– Comparing the results obtained from a new proposal and a baseline.

– Developing two projects in parallel ('twin projects') choosing one of them as the baseline.

– Applying the new proposal on some selected components and comparing the results obtained with the components that were not applied.

In this regard, we have decided to use the second strategy, which considers the development of twin projects (one project uses the traditional approach to manage a project while the other one implements our metamodel using MyPMP for Microsoft Project®). Therefore, the case study focuses on the application of the MyPMP add-in program in a small-sized software enterprise. Thus, the case study began with the definition of the scope of the pilot project included in the experiment. In accordance

with the objectives of this paper, we used the proposed add-in program to help project managers to define a process for managing a software project that fits with the necessities and resources (e.g., human, financial, equipment) of a small-sized software enterprise. By using MyPMP, an experimental group can load the assets of MoProSoft® and the PMBOK® Guide without altering the scope of the project, while the control group must use Microsoft® Project as usually. As we said, the processes defined by MyPMP are adapted according to the conceptual architecture of our metamodel that classifies the tasks, processes, products and measures to use in any project. Nevertheless, most of the obtained products, such as the requirements specification, follow the standards used within the small-sized software enterprise in order to minimize the change impact. The experimental and control groups were conformed by six people: one project manager, two analysts, two programmers, and one quality manager. The selection of the group's members was made based on their experience and knowledge about software development and project management, and their knowledge about the project's domain. All of the participants had the same number of years of experience in the enterprise (4 years).

Otherwise, the hypothesis tested in the study was as follows:

H1: The MyPMP add-in program can reduce the effort and time required to develop a new software project.

Additionally, in order to capture several aspects of performance related to MyPMP, it was necessary to define two response variables as follows:

- Effort. What percentage of the effort was reduced by applying correctly the proposed add-in program in project management?
- Time. What percentage of the estimated time for developing the software project was reduced by applying correctly the proposed add-in program?

To answer these questions, we examined the documentation of the project, observed participants during their work, applied questionnaires, and performed interviews. In this regard, the study has an explorative character.

4.1. Context of the study

The study was performed within a small-sized software enterprise, called SmallEnt for confidential reasons. SmallEnt uses regularly Microsoft® Project for planning and controlling all the software projects. The project developed in this case study was named 'MediControl', a software product developed for a small medical clinic. MediControl would allow doctors, patients, and nurses initiate and control the appointments, and manage the pharmaceutical products through their respective interfaces. Thus, MediControl would include the following modules: authentication of users, appointments management, cash flow, stock of pharmaceutical products, recipes generator, clients catalog, bar code generation for products, and reporting. In order to illustrate the application of MyPMP in the development of MediControl, Fig. 6 shows an

example of how the metamodel is implemented while being supported by the created add-in program.

4.2. Method

We collected the data in four different ways: review of historical documents, observation of participants, semi-structured interviews, and questionnaires using a Likert scale. It is noteworthy that the majority of the collected data were quantitative; nevertheless, also qualitative data were collected in lesser extent. During the review of historical documents, an expert researcher[2] reviewed if the assets (e.g., project plan, WBS, configuration management) were properly filled out. Otherwise, in relation to the observation of participants, it was necessary to participate in various planning and feedback meetings to get firsthand information concerning the problems about the project management. The interviews were applied to all team members of both groups.

Each interview began with general questions and discussions about how the project managers carry out the project management and the use of the add-in program to define and use an enacted process, in order to ensure that applicants had a common understanding of the concepts involved. Also, after using the add-in program, we applied a questionnaire using a Likert scale, to obtain the participants' opinions about the add-in program and its benefits.

4.3. Empirical results

As mentioned before, the created add-in program has provided a formal structure for defining customized project management processes within the context of a small-sized software enterprise. In this regard, the modules that implement the metamodel characteristics are accessible via a well-defined interface created for Microsoft® Project 2007. In order to assess the feasibility of the MyPMP add-in program, we have developed twin projects into a small-sized software organization. The results of this experiment are summarized in Table 1, in which column *Effort* shows the effort changed or extended, and column *Time* shows the approximate time spent in the management phase.

Table 1. Results from the Case Study

Control group		Experimental group		% of reduction	
Effort (hours)	Time (weeks)	Effort (hours)	Time (weeks)	Effort	Time
456	23	330	15	26.6	35

[2] This expert was not part of the research group, and he was unaware that his work was part of an experiment performed within a case study. Therefore, the expert made a completely blind evaluation on the both projects' documentation (obtained by the experimental and control groups), not knowing anything about how these data were obtained.

As we observed, the difference between the traditional approach (performed by the control group and using Microsoft Project® as usually, without MyPMP) to manage the software project and the proposed approach is relatively large.

Moreover, even unskilled project managers can work with MyPMP to define a process for managing the software projects; the developed add-in program provides them a strong support to define and simplify the management concept.

Furthermore, in the case study presented the difference between the control and experimental groups is evident. The effort required for the control group is about 75% more than the effort required for the experimental one; while the time required for the control group is about 65% more than the time required for the experimental one because MyPMP enables project managers to define a set of activities to develop a new software product. These results allow us to validate our hypothesis (H1: The MyPMP add-in program can reduce the effort and time required to develop a new software project).

Finally, the benefits offered by an add-in program for Microsoft Project® are achieved, while it is enriched with the specific functionalities required for incorporating the created metamodel.

4.4. Threats to validity

It is noteworthy that we did not use a statistical test over our hypothesis, because our case study only has focused on the analysis of the performance of two groups (the experimental group and the control group) in the development of a software project. Nevertheless, we have taking into account the advice of Runeson and Höst [45], defining the following main threats to validity the achieved results:

- Internal validity. In our case study, internal validity issues primarily deal with the causal issues of our findings. These concerns are addressed to some extent because of the fact that the participants had no knowledge that this study was being conducted to prevent the modification of their behavior/management practices and avoid affecting our measurements. Similarly, the participants (from the control and experimental groups) had the same experience in software development and project management, the same domain knowledge, and the same experience in number of years in the enterprise (4 years). Hence, there is no internal motivation to show results either way to influence this study; however, we cannot ensure that the employees participating in the experimental group have been particularly motivated by the use of the MyPMP add-in program or if this group had cleverer members than the control group.
- Construct validity. Construct validity issues arise when there are errors in measurement. In this sense, the process to collect data uses questionnaires (applied to the project managers and the rest of employees) using a well-structured Likert scale. Moreover, at the end of the project we performed interviews with the participants to assess if the projects fulfilled all the planned functionalities. Thus, we assure that independently from whom analyzes the data, the answer value for each person will be the same (passing from a qualitative approach to a quantitative one). However, we cannot discard that if the compared projects would have developed under and agile

approach, the control group would have had a more satisfactory product delivery than the experimental one.

– External validity. It is possible that some problems of external validity arise given the fact that the company where the case study was applied has a small size (no more than 15 employees, where 12 are devoted to the software development), if the set of software requirements was small, of the enterprise was very related to the project domain (i.e., management system).

5. Conclusions

Many project management methodologies have been developed from industry practices and international standards to ensure a higher rate of success for software projects. It is true that these methodologies have been widely and effectively used in large-sized software companies. However, when software projects are developed within the context of small-sized software enterprises, there is often a lack of an appropriate method for project management due to the unskilled project managers who cannot use the methodologies used in large-sized organizations. In fact, this situation is exacerbated when the current literature has determined that most of the problems in these organizations may be related to the lack of capacity to manage projects. Therefore, our study coincides with the findings of Sánchez-Gordón and O'Connor [46] in the sense that it is necessary to provide an insight toward a simplification of work products as they relate to the activities of the software development process in small-sized software enterprises for supporting the project success.

In this regard, this paper has presented a metamodeling approach for simplifying the main concepts related to project management and helping project managers to define a customized process in the context of a small-sized software enterprise. With this aim in mind, our study has initially analyzed MoProSoft®, for productive activities, and the PMBOK® Guide, for managerial activities, and we have then proposed an alternative metamodel that aims to provide a simplified view of both perspectives. This metamodel identifies the basic elements that enable small-sized software enterprises to focus their effort on improving at project-level and, in the future, at process-level. Thus, the definition of a customized process emphasizes the importance of individuals' skills to both develop and manage a software project and create a solid basis of knowledge to improve the software and product qualities. In order to support our approach, we have created the MyPMP add-in program for Microsoft Project® 2007, one of the most used management tools in the context of small-sized software enterprises, to evaluate the feasibility of our metamodel. We are currently collecting data through performing more case studies to provide more formal conclusions about the incorporation of the metamodel to define simplified processes in real projects within the context of small-sized software enterprises.

Finally, we have obtained some important lessons learned from the case study. However, there is still much to be done, especially if we focus on non-expert adoption; we learned a number of generic lessons that are helpful for similar situations. The most critical, in our opinion, are the following:

- Facilitate process definition: The MyPMP interfaces, which help to perform the process definition, are simple and easy to understand. Nevertheless, it was necessary to provide a solid background to all participants about the project management process because they lacked specific knowledge about it. It is important to mention that the lack of technical knowledge is one of the most important disadvantages within the context of small-sized software enterprises.
- Choice of vocabulary: It is important to encourage the creation of a vocabulary mapping (i.e., employees may be unaware of some terms used by the add-in program) thus reducing the overhead and need for expert intervention in the process.
- Long-term commitment: It is necessary the establishment of a solid commitment with top management and project managers of small-sized software enterprises to obtain better results.

References

1. Colomo-Palacios, R., Casado-Lumbreras, C., Soto-Acosta, P., García-Peñalvo, F. J., Tovar, E.: Project managers in global software development teams: a study of the effects on productivity and performance. Software Quality Journal, Vol. 22, No. 1, 3-19. (2014)
2. Mir, F. A., Pinnington, A. H.: Exploring the value of project management: Linking project management performance and project success. International Journal of Project Management, Vol. 32, No. 2, 202-217. (2013)
3. de Carvalho, M. M., Patah, L. A., de Souza, B. D.: Project management and its effects on project success: Cross-country and cross-industry comparisons. International Journal of Project Management, Vol. 33, No. 7, 1509-1522. (2015)
4. Cerpa, N., Bardeen, M., Astudillo, C. A., Verner, J.: Evaluating different families of prediction methods for estimating software project outcomes. Journal of Systems and Software, Vol. 112, 48-64. (2016)
5. Turner, R., Ledwith, A., Kelly, J.: Project management in small to medium-sized enterprises: Matching processes to the nature of the firm. International Journal of Project Management, Vol. 28, No. 8, 744-755. (2010)
6. O'Connor, R. V., Laporte, C. Y.: Software project management in very small entities with ISO/IEC 29110. In: Winkler, D., O'Connor, R. V., Messnarz, R. (eds.): Systems, Software and Services Process Improvement. Communications in Computer and Information Science, Vol. 301. Springer-Verlag, Berlin Heidelberg, 330-341. (2012)
7. Caniëls, M. C. J., Bakens, R. J. J. M.: The effects of project management information systems on decision making in a multi project environment. International Journal of Project Management, Vol. 30, No. 2, 162-175. (2012)
8. Turner, R., Ledwith, A., Kelly, J.: Project management in small to medium-sized enterprises: Tailoring the practices to the size of company. Management Decision, Vol. 50, No. 5, 942-957. (2012)
9. Richardson, I.: Why are small software organizations different? IEEE Software, Vol. 24, No. 1, 18–22. (2007)
10. Henderson-Sellers, B.: Bridging metamodels and ontologies in software engineering. Journal of Systems and Software, Vol. 84, No. 2, 301-313. (2011)
11. Callegari, D. A., Bastos, R. M.: Project management and software development processes: integrating RUP and PMBOK. In Proceedings of the 2007 International Conference on Systems Engineering and Modeling (ICSEM '07), Haifa, Israel, 1–8. (2007)

12. Alba, E., Chicano, J. F.: Software project management with GAs. Information Sciences, Vol. 177, No. 11, 2380-2401. (2007)
13. Chen, W. N., Zhang, J.: Ant colony optimization for software project scheduling and staffing with an event-based scheduler. IEEE Transactions on Software Engineering, Vol. 39, No. 1, 1-17. (2013)
14. Lee, S., Yong, H. S.: Distributed agile: project management in a global environment. Empirical Software Engineering, Vol. 15, No. 2, 204-217. (2010)
15. Persson, J. S., Mathiassen, L., Aaen, I.: Agile distributed software development: enacting control through media and context. Information Systems Journal, Vol. 22, No. 6, 411-433. (2012)
16. Petter, S., Vaishnavi, V.: Facilitating experience reuse among software project managers. Information Sciences, Vol. 178, No. 7, 1783-1802. (2008)
17. Gasik, S.: A model of project knowledge management. Project Management Journal, Vol. 42, No. 3, 23-44. (2011)
18. Kellner, M. L.: Software process modeling support for management planning and control. In Proceedings of the First International Conference on the Software Process. Redondo Beach, California, USA, 8-28. (1991)
19. Bryde, D. J.: Modelling project management performance. International Journal of Quality & Reliability Management, Vol. 20, No. 2, 229-254. (2003)
20. Ståhl, D., Bosch, J.: Modeling continuous integration practice differences in industry software development. Journal of Systems and Software, Vol. 87, No. 1, 48-59. (2014)
21. Cerón, R., Dueñas, J. C., Serrano, E., Capilla, R.: A meta-model for requirements engineering in system family context for software process improvement using CMMI. In: Bomarius, F., Komi-Sirviö, S. (eds.): Product Focused Software Process Improvement. Lecture Notes in Computer Science, Vol. 3547. Springer-Verlag, Berlin Heidelberg, 173-188. (2005)
22. Fernández, D. M., Penzenstadler, B., Kuhrmann, M., Broy, M.: A meta model for artefact-orientation: fundamentals and lessons learned in requirements engineering. In: Petriu, D., Rouquette, N., Haugen, Ø. (eds.): Model Driven Engineering Languages and Systems. Lecture Notes in Computer Science, Vol. 6395. Springer-Verlag, Berlin Heidelberg, 183-197. (2010)
23. Goknil, A., Kurtev, I., van de Berg, K., Spijkerman, W.: Change impact analysis for requirements: A metamodeling approach. Information and Software Technology, Vol. 56, No. 8, 950-972. (2014)
24. Henderson-Sellers, B., Gonzalez-Perez, C.: A comparison of four process metamodels and the creation of a new generic standard. Information and Software Technology, Vol. 47, No. 1, 49-65. (2005)
25. Ayed, H., Vanderose, B., Habra, N.: A metamodel-based approach for customizing and assessing agile methods. In Proceedings of the 8th International Conference on the Quality of Information and Communications Technology (QUATIC), Lisbon, Portugal, 66-74. (2012)
26. García, F., Serrano, M., Cruz-Lemus, J., Ruiz, F., Piattini, M.: Managing software process measurement: A metamodel-based approach. Information Sciences, Vol. 177, No. 12, 2570-2586. (2007)
27. Colombo, A., Damiani, E., Frati, F., Ontolina, S., Reed, K., Ruffatti, G.: The use of a meta-model to support multi-project process measurement. In Proceedings of the 15th Asia-Pacific Software Engineering Conference (APSEC '08), Beijing, China, 503-510. (2008)
28. Martins, P. V., da Silva, A. R.: PIT-ProcessM: A software process improvement meta-model. In Proceedings of the 7th International Conference on the Quality of Information and Communications Technology (QUATIC), Porto, Portugal, 453-458. (2010)
29. Tian, L., Zeng, G. Y., Yu, L., Zhu, B.: Research and implementation of software process metamodel for CMMI. Computer Engineering Design, Vol. 18, 245-267. (2010)

30. Banhesse, E. L., Salviano, C. F., Jino, M.: Towards a metamodel for integrating multiple models for process improvement. In Proceedings of the 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), Izmir, Turkey, 315-318. (2012)
31. Abels, S., Ahlemann, F., Hahn, A., Hausmann, K., Strickmann, J.: PROMONT —a project management ontology as a reference for virtual project organizations. In: Meersman, R., Tari, Z., Herrero, P. (eds.): On the Move to Meaningful Internet Systems. Lecture Notes in Computer Science, Vol. 4277. Springer-Verlag, Berlin Heidelberg, 813-823. (2006)
32. Velić, M., Padavić, I., Dobrović, Ž.: Metamodel of agile project management and the process of building with LEGO® bricks. In Proceedings of the 23rd Central European Conference on Information and Intelligent Systems (CECIIS), Varazdin, Croatia, 481–193. (2012)
33. Thiemich, C., Puhlmann, F.: An agile BPM project methodology. In: Daniel, F., Wang, J., Weber, B. (eds.): Business Process Management. Lecture Notes in Computer Science, Vol. 8094. Springer-Verlag, Berlin Heidelberg, 291-306. (2013)
34. Beydoun, G., Low, G., Henderson-Sellers, B., Mouratidis, H., Gomez-Sanz, J., Pavon, J., Gonzalez-Perez, C.: FAML: A generic metamodel for MAS development. IEEE Transactions on Software Engineering, Vol. 35, No. 6, 841-863. (2009)
35. Othman, S. H., Beydoun, G.: Metamodelling approach to support disaster management knowledge sharing. In Proceedings of the 21st Australasian Conference on Information Systems (ACIS), Atlanta, Georgia, USA, 1-10. (2010)
36. Atkinson, C.: Metamodelling for distributed object environments. In Proceedings of the First International Enterprise Distributed Object Computing Workshop (EDOC'97), Gold Coast, Queensland, Australia, 90-101. (1997)
37. International Organization for Standardization/International Electrotechnical Commission.: ISO/IEC 24744. Software Engineering—Metamodel for Development Methodologies. ISO, Geneva. (2007)
38. Gonzalez-Perez, C., Henderson-Sellers, B.: Modelling software development methodologies: a conceptual foundation. Journal of Systems and Software, Vol. 80, No. 11, 1778-1796. (2007)
39. Object Management Group. Software & Systems Process Engineering Meta-Model Specification, Version 2.0. Available at: http://doc.omg.org/formal/08-04-01.pdf (2008)
40. Mesquida, A. L., Mas, A.: A project management improvement program according to ISO/IEC 29110 and PMBOK®. Journal of Software: Evolution and Process. Vol. 26, No. 9, 846-854. (2014)
41. Oktaba, H..: MoProSoft: A software process model for small enterprises. In Proceedings of the First International Research Workshop for Process Improvement in Small Settings, Pittsburgh, Pennsylvania, USA, 93-100. (2005)
42. Lu, X., Liu, H., Ye, W.: Analysis failure factors for small & medium software projects based on PLS method. In Proceedings of the 2nd IEEE International Conference on Information Management and Engineering (ICIME), Chengdu, China, 676-680. (2010)
43. Kitchenham, B., Pickard, L., Pfleeger, S. L.: Case studies for method and tool evaluation. IEEE Software, Vol. 12, No. 4, 52-62. (1995)
44. Wohlin, C., Höst, M., Henningsson, K.: Empirical research methods in software engineering. In: Conradi, D., Wang, A. I. (eds.): Empirical Methods and Studies in Software Engineering. Lectures Notes in Computer Science, Vol. 2765. Springer-Verlag, Berlin Heidelberg, 7-23. (2003)
45. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering, Vol. 14, No. 2, 131-164. (2009)
46. Sanchéz-Gordón, M. L., O'Connor, R. V.: Understanding the gap between software process practices and actual practice in very small companies. Software Quality Journal, Vol. 24, No. 3, 549-570. 2016

Ivan Garcia holds a PhD in Software and Systems from the Universidad Politécnica de Madrid and he is full-time professor at the Division de Estudios de Posgrado of the Universidad Tecnológica de la Mixteca, Mexico. He is author of international papers related to Software Engineering and, more specifically, Software Process Improvement. He also is a member of the team that has translated CMMI-DEV V1.2 and CMMI-DEV v1.3 to Spanish.

Carla Pacheco studied a PhD in Informatic Language and Software Engineering in the Universidad Politécnica de Madrid and she is full-time professor at the Division de Estudios de Posgrado of the Universidad Tecnológica de la Mixteca, Mexico. She is author of international papers related to Software Engineering and, more specifically, Requirements Engineering. She also has participated in several Mexican projects related to software requirements as a part of process to improve the Mexican software industry.

Magdalena Arcilla-Cobián holds a PhD in Computer Science. She is an assistant professor in the Computer Science School at the Universidad Nacional de Educación a Distancia (Open University). She is teaching in the area of Software Engineering, specifically in the domain of software process management and improvement. She is author of papers related to process improvement, mainly in the service domain. She holds the ITIL® v2 and v3 Foundation certificates.

Jose A. Calvo-Manzano holds a PhD in Computer Science. He is an assistant professor in the Computer Science School at the Universidad Politécnica de Madrid. He is teaching in the area of Software Engineering, specifically in the domain of software process management and improvement. He has participated in more than 20 research projects (European and Spanish Public Administration). He is author of more than 50 international papers. He is author of books related to software process improvement and software engineering topics also. He has been a member of the team that has translated CMMI-DEV v1.2 and CMMI-DEV v1.3 to Spanish. He also holds the ITIL® v2 and v3 Foundation, and CMDB Certification.

Received: August 1, 2016; Accepted: October 20, 2016.