Journal of Software: Evolution and Process

JOURNAL OF SOFTWARE MAINTENANCE AND EVOLUTION: RESEARCH AND PRACTICE
J. Softw. Maint. Evol.: Res. Pract. 2012; **24**:851–876
Published online 23 May 2011 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/smr.541

A software maintenance methodology for small organizations: Agile_MANTEMA

Francisco J. Pino[1,*,†], Francisco Ruiz[2], Félix García[2] and Mario Piattini[2]

[1]*IDIS Research Group, Electronic and Telecommunications Engineering Faculty, University of Cauca, Calle 5 #4-70 Popayán, Colombia*
[2]*ALARCOS Research Group, Information Systems and Technologies Department, UCLM–INDRA Research and Development Institute, University of Castilla–La Mancha, Paseo de la Universidad, 4-13071 Ciudad Real, Spain*

SUMMARY

Software maintenance is an important, indeed vital, activity for software companies. For many organizations, including small firms, it is also a business opportunity. In the quest to help companies of this type, for whom strengthening their maintenance process is a chief concern, we present Agile_MANTEMA. This is a methodological proposal for software maintenance, one which focuses on small companies. The methodology defines an agile maintenance strategy, setting out in detail what is to be carried out, when, how and by whom; that is to say, it seeks to give a detailed guide for the implementation of the maintenance process in these kinds of companies. Agile_MANTEMA also establishes a set of elements, such as types of maintenance, service levels and capability levels, which aim to: (i) handle the complexity that is inherent to the maintenance process and (ii) allow a small company to define its own maintenance process, taking into account its particular characteristics and needs. In an effort to improve their maintenance process, two small firms conducted an improvement project in which Agile_MANTEMA was used. The experiences reported by these companies tell us that this methodology offers a strategy that can be useful, practical and suitable for small companies as they tackle software maintenance. Copyright © 2011 John Wiley & Sons, Ltd.

Received 21 January 2010; Revised 25 February 2011; Accepted 8 March 2011

KEY WORDS: software maintenance; MANTEMA methodology; small software organizations; scrum; COMPETISOFT; experience report

1. INTRODUCTION

There are studies [1–5] that point to software maintenance as being the most costly part of the software life cycle. They also find that the percentage of resources needed for maintenance increases proportionately as more software is produced. Furthermore, according to Singer [6], 61% of the professional life of programmers is dedicated to maintenance and only 39% to new developments, a statement that is still valid today. According to the exhaustive longitudinal study on software development and maintenance discussed in [7], the maintenance work done to improve systems in operation varies from between 53 and 73%. In [8], the specific case of the Latin American software scene is analysed. This shows that the higher the proportion between the software maintenance contracts and the total company sales volume, the lower the number of new customers. The inverse case also appears, i.e., the smaller the proportion between maintenance contracts and total sales

*Correspondence to: Francisco J. Pino, IDIS Research Group, Electronic and Telecommunications Engineering Faculty, University of Cauca, Calle 5 #4-70 Popayán, Colombia.
†E-mail: fjpino@unicauca.edu.co

volume, the greater the quantity of new clients in the company's books. These are some of the elements which bring out just how important and indeed, how critical, software maintenance is for the companies that make up the software industry. Many software organizations do not have processes defined for their software maintenance activities [9], however. At present there is still a lack of: (i) process management in software maintenance (according to Van Bon [10]) and (ii) maintenance process models (according to Kajko-Mattsson [3]).

At the same time, it is a well-known fact that the software industry is made of mostly small organizations (firms with fewer than 50 employees). The chaotic model of software development in numerous organizations of this kind [11], along with market pressures, means that it is quite normal for these companies to invest a great deal of effort in maintenance activities for liberated software, with insufficient quality assurance. It is also the case that some of these organizations want to take advantage of software maintenance as a business opportunity, by specializing in maintenance of software products developed by other organizations, i.e., by offering an external maintenance service (outsourcing).

The state of affairs outlined in the paragraphs above leads us to affirm that it is important to offer small organizations with methods and guides which will allow them to carry out and manage software maintenance activities appropriately. A proposal of this type is needed because: (i) software development methods are not suitable for driving activities related to software maintenance [12], (ii) some of the proposals focusing on software maintenance are heavyweight approaches and are not appropriate for small organizations (such proposals include the Model to improve the maintenance of software [13], SMmm [9] or MANTEMA [14, 15] among others) and (iii) other proposals, such as the corrective maintenance maturity model—CM3 [3], could be used by these kinds of organizations, but are only for a particular type of software maintenance. Furthermore, from the demands expressed by several small companies participating in the COMPETISOFT Latin-American project [16], there arises the need for a proposal which would focus on the overall software maintenance and which would be tailored to the characteristics of this category of organizations. This proposal would have to take into account the software maintenance strategies described by existing maintenance methodologies. These must be adapted to make them lighter (or agile), so that they can be useful and practical for small organizations.

Bearing in mind the scenario described above, this paper presents Agile_MANTEMA, a methodological proposal for software maintenance which focuses on small organizations. It is based on making the process elements described by MANTEMA lighter and on the incorporation of agile project management by using the Scrum method. This proposal defines a maintenance strategy that sets out in detail what should be carried out, when, how and by whom. That is to say, it seeks to guide the software maintenance process for these kinds of organizations, step-by-step. The paper also describes our experience of the application of the proposed maintenance methodology in two small software companies. These companies used Agile_MANTEMA to improve their software maintenance processes. The initial results obtained from the experience reports lead us to observe that this methodology can be useful, practical and suitable for supporting software maintenance in these types of organizations.

The remainder of this paper is organized as follows: Section 2 describes the background, the related works and the research method of Agile_MANTEMA. An overview of the elements described by this maintenance methodology is given in Section 3. Section 4 provides an explanation of the maintenance process proposed by Agile_MANTEMA and in Section 5 there is a description of the application of this methodology in two small companies. Finally, we present the conclusions and the future work.

2. BACKGROUND

Agile_MANTEMA has been developed in the context of the COMPETISOFT project [16]. This project was funded by the 'Programa Iberoamericano de Ciencia y Tecnología para el Desarrollo', CYTED (Ibero-American Science and Technology Development Programme), which involved

DOI: 10.1002/smr

1 national body for standardization and certification, more than 10 small software companies and 27 research groups from 13 countries in Latin America [17]. The software process improvement strategy proposed by this project pursues an increase in the level of competitiveness of Latin-American small software organizations. Based on the expectations and requirements from the companies and researchers, COMPETISOFT considers software maintenance to be a highly important activity that should be tackled separately from software development. This is because the nature and characteristics of each are very different; many techniques, tools, model processes, etc., from development are not directly applicable to maintenance [12]. Indeed, many small organizations have to develop pure software maintenance projects, which makes it very important for them to apply specific maintenance methodologies. In this regard, the COMPETISOFT approach has developed a software maintenance methodology which adapts the MANTEMA methodology [14] and Scrum method [18] to the special characteristics of small organizations. This methodology is Agile_MANTEMA, which is created from a speeding up of MANTEMA by means of project agile management, as proposed by Scrum.

MANTEMA was developed with medium and large maintenance projects and services in mind, especially outsourced ones, as, for example those to do with banks or public administration. That is why one of Mantema's major characteristics is that it follows a maintenance service that is based on the ISO 12207:2002 standard [19], with a very elaborate definition. This definition contains: a flow of activities and tasks to be performed, a model of roles and participating organizations, inputs and outputs to and from the activities and a list of techniques recommended for each activity, along with a review of the types of maintenance. All the above components were derived from experience in medium- and large-scale projects. That being so, in the study presented in [14], a clear difference has been established between the workflow which is to take place in the urgent corrective maintenance (non-plannable) and the remaining kinds of maintenance (the plannables). An extensive description of the development and validation of this methodology is described in [12], which shows that the maintenance strategy of MANTEMA is appropriate and that it can bring benefits to the companies using it.

On the other hand, agile methods, with their advocacy of self-empowered teams and shared ownership, are associated more with the management style of small organizations [20]. This is because this type of company considers that agile practices can be applied in their software process. That is due largely to the small initial investment required and because those practices allow them to take advantage of competitiveness in their personnel [21]. As [22] points out, some agile methods that focus on small teams are: Agile modelling [23], Extreme programming [24], Pragmatic programming [25] and Scrum [18]. Of all these methods, however, only the Scrum method provides a framework for managing projects, whereas the others focus on describing practices, activities and work product techniques to do with software development. That is to say, Scrum emphasizes management values and practices, without including practices on technical issues. For this reason, Scrum is a framework that can be integrated with various methods, processes and techniques [26]. This feature allows us to combine the framework for managing projects, as described by Scrum, with technical processes of a specific knowledge area (for instance, MANTEMA, from the software maintenance area) and thus offer them the possibility of more agile management. Bearing in mind the aspects described previously, the incremental and iterative process proposed by Scrum has been integrated and tailored to the MANTEMA software maintenance methodology, to create Agile_MANTEMA.

Table I presents a comparison between MANTEMA and Agile_MANTEMA, to show a clear distinction between these two methodologies.

We have put the term 'agile' into the name of this methodology, because the use of Scrum allows the proposed methodology to tackle some important aspects. We can consider the following advantages: (i) the maintainers are given a technical and managerial maintenance environment to support their daily work, and are trusted to get the job done, (ii) there is guaranteed early and continuous delivery of the maintained software to the customer, (iii) frequent delivery of maintained software takes place within a short timescale, (iv) close work and communication occurs between maintainers and users throughout the maintenance project and (v) reflection about the process

DOI: 10.1002/smr

Table I. Comparison between Agile_MANTEMA and MANTEMA methodologies.

	Agile_MANTEMA	MANTEMA
Number of roles	5	8
Number of activities	10	14
Number of tasks	27	46
Number of work products	3	11
Number of proposed metrics	1	11
Management style	Agile and decentralized based on Scrum	Heavier and centralize
Maintenance process	Incremental and interactive, less controlled, adaptive, cycles numerous	Stiffer, more controlled, predictive, cycles limited
Perspective on change	Adaptability to changeable modification requests	Some resistance to changeable modification requests
Maintenance client/user	Member of the maintenance team	Involvement with the maintenance team
Delivery of maintained software	Early in the maintenance project	Later on in the maintenance project
Documentation	Low	Heavier
Service levels	Yes	No
Process performance levels	Yes	No
Process capability levels	Yes	No

happens on an ongoing basis, along with continuous learning of the maintenance work. This refines and improves the daily activity. These aspects are related to several principles described by the agile manifesto [27].

2.1. Related work

We are not aware of any other attempts to provide an explicit and comprehensive software maintenance methodology which focuses on small organizations. In [9] a software maintenance maturity model is discussed, aiming to address the assessment and improvement of the software maintenance function. The validation of this maturity model has shown that there is a negative perception of its usefulness in small organizations, however, since it has too many practices for it to be used by this type of a company. Business experiences of the implementation of extreme programming practices in real-software maintenance projects are presented in [28, 29]. An adoption of the extreme programming practices for carrying out maintenance activities is discussed in [28]. The results of introducing extreme programming in an evolutionary and maintenance software development environment are presented in [29]. This study finds that many of the extreme programming practices have to be modified if they are to work in maintenance environments. That is to say, these last two studies focus on adapting the extreme programming practices and/or principles for the purpose of carrying out practices of software maintenance. These proposals do not describe explicitly how to manage and lead the activities and practices involved in the software maintenance, however. Neither do they consider other activities and practices that are unique to, and fundamental for, software maintenance.

A proposal that deals with software maintenance in great detail is the corrective maintenance maturity model—CM3, which has been presented and discussed at length in [3, 30–35]. CM3 is a process model for handling explicitly the maintenance category related to corrective maintenance. This model consists of several processes (which perform a clearly defined task) collaborating with each other. Each process has a well-defined structure, providing detailed guidance to the organizations when building or improving their maintenance processes [31]. This model is a very comprehensive approach for corrective maintenance, but it was not developed with the particular characteristics of small organizations in mind. That could make it difficult to apply in these types of organizations. Furthermore, this model focuses only on a specific maintenance type; we believe that small organizations need a proposal that allows them to address all the maintenance types

DOI: 10.1002/smr

suitably. In this sense, it is important to highlight that, in the real world, clients' maintenance requests are not filtered previously by maintenance type before they reach us.

It should be mentioned that there are several studies related to the software process in small organizations (such as MESOPyME [36], IMPACT [37], ASPE-MSC [38], MoproSoft [39], MPS.BR [40], ADEPT [41], RAPID [42], PROCESSUS [43], among others). None of these proposes the definition of a maintenance methodology or process for small companies, however. Some of the studies do propose process reference models that present a set of processes which small software organizations could use to derive significant benefit during the software development and maintenance of a software product. These are, for instance:

- MoProSoft proposes 6 processes (based on ISO 12207, CMM, ISO 9001): Business management, Process management, Project management, Resource management, Administration of specific projects and Software development and maintenance.
- MPS.BR proposes 23 processes (based on ISO 12207 and CMMI): Organizational innovation and deployment, Causal analysis and resolution, Organizational process performance, Quantitative project management, Risk management, Decision analysis and resolution, Requirement development, Technical solution, Validation, Verification, Software integration, Software installation, Product release, Training, Process establishment, Process assessment and improvement, a Tailoring process for project management, Configuration management, Quality assurance, Acquisition, Measurement, Project management and Requirement management.
- ADEPT proposes 12 processes (based on CMMI): Requirement management, Configuration management, Project planning, Project monitoring & control, Measurement & analysis, Process & product quality assurance, Risk management, Technical solution, Verification, Validation, Requirement development, Product integration.
- RAPID proposes 8 processes (based on ISO 15504:1998): Requirement elicitation, Software development, Configuration management, Quality assurance, Project management, Problem resolution, Risk management, Process establishment.
- PROCESSUS proposes 6 processes (based on CMM and ISO 9001): Customer relationship management, Project management, Software engineering, Supporting activities, Process management and Process automation.

Of all these process reference models for small software organizations, only MoProSoft makes mention of 'maintenance', but the activities described in this process are related more to software development than to software maintenance. It should also be pointed out that none of these reference models provides a detailed approach for supporting the implementation of software maintenance.

The contribution of the proposal described in this paper is to present Agile_MANTEMA, a software maintenance methodology that is explicitly for small software organizations. This methodology takes into account: (i) the specific software maintenance activities and the different maintenance types (since it is based on MANTEMA), along with (ii) the features of the small software organization. That being so, we have analysed and selected the activities from MANTEMA that are suitable for these types of organizations. Moreover, in Agile_MANTEMA the Scrum method has been integrated, aiming to give detailed guidelines for supporting the management and performance of these software maintenance activities. As discussed previously, our reason for doing so is that this method provides support for project management and it is suitable for small teams. The purpose is to provide small software organizations with a strategy that can be useful and practical for the execution of the software maintenance process. Moreover, if a small organization employs this strategy it can: (i) manage the complexity of the maintenance process and (ii) define its own process of maintenance, taking into account its particular characteristics and needs.

2.2. Research method

We have used the action-research method to define, refine and apply the Agile_MANTEMA methodology. According to McKay and Marshall [44] and Chiasson *et al.* [45], action-research involves a research cycle and a problem-solving cycle, in which the knowledge is applied and

DOI: 10.1002/smr

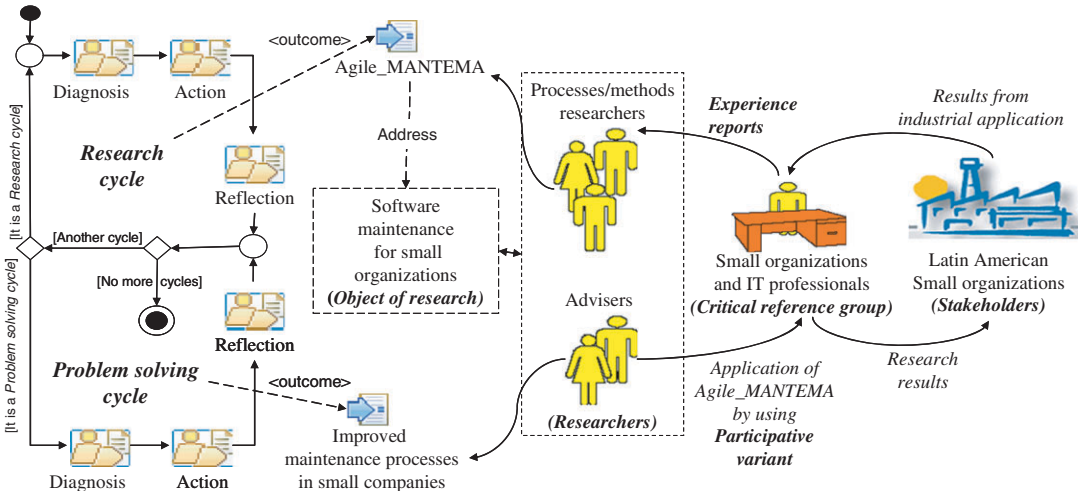

Figure 1. Research strategy used to develop and apply Agile_MANTEMA.

discovered interactively between activities with different goals and outcomes. Each of these cycles includes at least the following activities: problem diagnosis, action intervention, and reflective learning [46]. Bearing in mind these aspects, Figure 1 shows a high-level view of the research strategy that we followed. In our case, we began the research cycles which were carried out with the goal of developing an initial version of the Agile_MANTEMA. This could be applied by the small organizations participating in the COMPETISOFT project, by means of problem-solving cycles. The information and knowledge acquired from the problem-solving cycle was registered in the respective experience reports. This knowledge was used in the following research cycles to refine and improve the Agile_MANTEMA methodology, thereby creating a new version of this.

Figure 1 also shows the elements that should be considered in action-research, according to Wadsworth [47]. These elements are: (i) the researched object, (ii) researchers, (iii) critical reference group and (iv) stakeholders of the research. By using the action-research method, the participants involved in the development of Agile_MANTEMA were divided into two groups: the first was made of *researchers* from various universities; the second is called the *critical reference group*, and it included IT professionals from small software organizations. The researcher group was divided into two: (i) *processes/methods researchers*, who were responsible for developing the components of Agile_MANTEMA and (ii) *advisers*, who were the field researchers responsible for carrying out the application of this maintenance methodology in the organizations of the *critical reference group*. As mentioned previously, the application of action-research is iterative and has allowed us to have continual feedback between the *researchers* and the companies involved, hence we can polish up the components of the proposed maintenance methodology. With regard to the execution of the action intervention activity of the problem-solving cycle from action-research, the *researchers* applied the participative variant [48]. That is to say, the application of Agile_MANTEMA was participative, because the *critical reference group* put this methodology into practice with support from an *adviser*. That group shared its experiences, effects and results of the application with the *researchers* by means of experience reports (see Figure 1).

3. OVERVIEW OF AGILE_MANTEMA

Agile_MANTEMA offers small organizations a methodological guide for conducting the software maintenance process. This methodology describes:

- A process, which is a detailed guide for managing and carrying out software maintenance step-by-step. This process describes activities that support the different types of maintenance (urgent corrective, non-urgent corrective, perfective, preventive and adaptive).

DOI: 10.1002/smr

- Service levels, by which the type of maintenance offered at each service level is established, along with the basic auxiliary processes which support this type of maintenance. A service level defines the scope (maintenance types and auxiliary processes) of the maintenance service. The auxiliary processes describe support and management practices which can strengthen the performance of the maintenance process.
- Process performance levels, through which basic auxiliary processes of support and management are grouped, in an attempt to offer an orderly and scalable way to strengthen the carrying out of the maintenance process.
- Process capability levels, by means of which the process capability involved in the software maintenance is reinforced.

The overall structure of the process for software maintenance, as set out in Agile_MANTEMA, is displayed in Figure 2. The general activities defined by this process are: definition of the maintenance process, record and analysis of modifications requests, execution of the intervention (these last two activities are cyclical) and migration and retirement of the software. This general structure is based on the maintenance process of MANTEMA and on the final description of the maintenance process defined in the ISO 12207:2008 standard [49]. Unlike those approaches, however, this macrostructure proposes and includes the carrying out of the software maintenance in an iterative and incremental way, by means of the definition of the cycle, made of the following activities: Record and analysis of modification requests and Execution of the intervention.

This maintenance methodology is complemented by the following elements which have been analysed and adapted so that they may be integrated into Agile_MANTEMA: (i) Service level, extracted from Métrica V3 [50], (ii) Process performance, based on the ISO/IEC 15504-5 standard [51] and (iii) Process capability, based on the ISO/IEC 15504-2 [52] standard. The concepts of service level and process performance are integrated into Agile_MANTEMA and are related within it in a two-dimensional way. Thus, on one dimension the service levels are found and on the other the performance level is seen (see Figure 3). In Section 3.3 there is a description of how these two concepts are related to process capability.

The purpose of offering this two-dimensional representation is that any small organization can handle the complexity inherent in the software maintenance process. It does so through adaptation and implementation of this process, in line with its own organizational characteristics and business goals. The intention is for a small company to be able to choose the service level it wishes to offer in a particular project or service, along with the performance level and the capability level that it wants to implement in its own maintenance process, according to its particular needs and infrastructure.

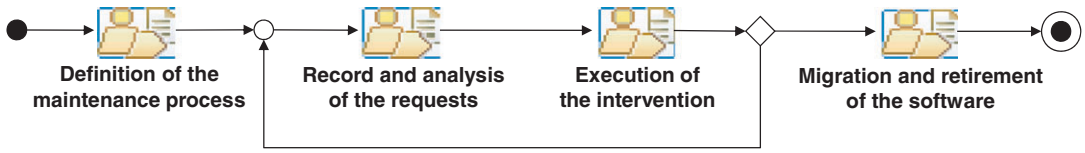

Figure 2. General structure of the maintenance process described by Agile_MANTEMA.

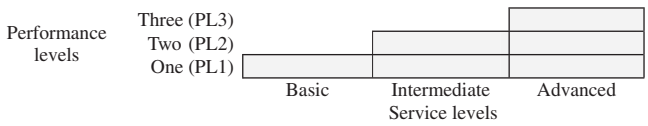

Figure 3. Two-dimensional representation of Agile_MANTEMA.

DOI: 10.1002/smr

Table II. Service levels of Agile_MANTEMA.

	Basic	Intermediate	Advanced
Maintenance Types supported by the Agile_MANTEMA process	• Urgent corrective	• Urgent corrective • Non-urgent corrective • Perfective	• Urgent corrective • Non-urgent corrective • Perfective • Preventive • Adaptive
Basic interfaces with other auxiliary support and management processes	OPE.2 Customer support (PL1) SUP.9 Problem resolution management (PL1)	OPE.2 Customer support (PL1) SUP.9 Problem resolution management (PL1) SUP.8 Configuration Management (PL2) SUP.1 Quality Assurance (PL2)	OPE.2 Customer support (PL1) SUP.9 Problem resolution management (PL1) SUP.8 Configuration Management (PL2) SUP.1 Quality Assurance (PL2) SUP.10 Change request Management (PL3) MAN.3 Project management (PL3)

3.1. Maintenance types and service levels

The types of maintenance in Agile_MANTEMA are the five kinds that are identified in MANTEMA [12, 53], since this is not a factor that is affected by the search for greater agility. These maintenance types are: (i) Urgent corrective: a detected error prevents normal system operation and the solution time is critical, (ii) Non-urgent corrective: a detected error does not block the normal operation of the system and the solution time is not critical, (iii) Perfective: when new functionalities are added to the system, (iv) Preventive: consists of the software modification to improve its maintainability and quality properties and (v) Adaptive: when the system will change its execution environment. The types of maintenance are organized into categories of 'plannable' and 'non-plannable'. The urgent corrective maintenance is non-plannable and is similar to the emergency maintenance proposed in [34], while the maintenance types which are non-urgent corrective, perfective, adaptive and preventative are plannable. For instance, an operational block of the software product that prevents it working properly should be addressed as non-plannable maintenance, whereas a new requirement that comes from the change in the laws and regulations should be dealt with as plannable maintenance. The division of the maintenance into these categories allows better management. It also ensures optimization of the ordered group of modification requests, thereby offering a criterion for classifying and prioritizing the requests towards the particular role in charge of carrying out that task.

Agile_MANTEMA establishes three service levels (basic, intermediate and advanced), identifying the types of maintenance which are given at each level (see Table II). In addition, it also defines the interfaces which each service level has with the other auxiliary support and management processes. The description of each process can be obtained from the ISO/IEC 15504-5 [51] standard.

The implementation of the activities described by the auxiliary processes (based on the interfaces of the service levels described in Table II), makes it possible to carry out base practices which reinforce the maintenance process defined by Agile_MANTEMA. The auxiliary processes are made of base practices and work products, which are software engineering activities that directly guide the purpose of each particular process. These activities contribute to the generation of the process outputs, which are useful for software maintenance. To incorporate the practices described in the

DOI: 10.1002/smr

Table III. Support for the auxiliary processes on the basis of the works analysed.

	Selection supported by			
Auxiliary process	[51]	[9]	[54]	[51]
OPE.2 Customer support		X		
SUP.9 Problem resolution management	X	X		X
SUP.8 Configuration management		X	X	X
SUP.1 Quality assurance		X	X	X
SUP.10 Change request management	X	X	X	X
MAN.3 Project management			X	X

auxiliary processes in the maintenance process, we have used and adapted the concept of interfaces presented in Métrica V3 [50]. Section 4.3 shows an example of an interface.

For the choice of auxiliary processes described in Table II we have, on the other hand, borne in mind the following pieces of work: (i) the description of the maintenance process presented in the ISO/IEC 15504-5 standard [51], (ii) the process areas and KPAs of the Software maintenance maturity model presented in [9], (iii) the processes that are considered critical to the implementation of a process improvement project in small software organizations and are therefore of priority, as presented in [54] and (iv) the related processes for process attributes of level two that are discussed in the standard ISO/IEC 15504-5 [51] (since this is the level that a small organization wishes to reach initially [55]). This work has been analysed, in an attempt to establish what the most appropriate auxiliary support and management processes are, for software maintenance, as well as for small organizations. Table III shows each of the auxiliary processes defined for Agile_MANTEMA, along with the relationship between each of these processes and the publications supporting them.

3.2. Performance levels

The performance levels establish what auxiliary processes should be defined and put into practice to strengthen the maintenance process by means of support and management practices. In this sense, Agile_MANTEMA defines three performance levels:

- Performance level one (PL1), which establishes that the auxiliary processes to be implemented are OPE.2 (Customer support) and SUP.9 (Problem resolution management).
- Performance level two (PL2), which establishes that the auxiliary processes to be implemented are those corresponding to level 1 plus SUP.8 (Configuration Management) and SUP.1 (Quality Assurance).
- Performance level three (PL3), which establishes that the auxiliary processes to be implemented are all indicated on two previous levels, plus SUP.10 (Change request Management) and MAN.3 (Project management).

Taking into account Figure 3 and Table II, we should remark at this point that Agile_MANTEMA establishes the following performance levels for each of the service levels:

- 'Basic' has just one performance level (PL1), which means that the activities described in (i) the Agile_MANTEMA maintenance process which support the type of maintenance defined for this service level (Urgent corrective), and (ii) the related auxiliary processes (OPE.2 and SUP.9) should be carried out.
- 'Intermediate' has two different performance levels (PL1 and PL2), which means that the activities described in the following processes should be carried out; these are: (i) The Agile_MANTEMA maintenance process which supports the maintenance types Urgent corrective, Non-urgent corrective and Perfective, and (ii) The related auxiliary process. We bear in mind that if only the OPE.2 and SUP.9 activities are performed, then the performance level is PL1, whereas if the SUP.8 and SUP.1 processes are also carried out, the performance level is PL2.

DOI: 10.1002/smr

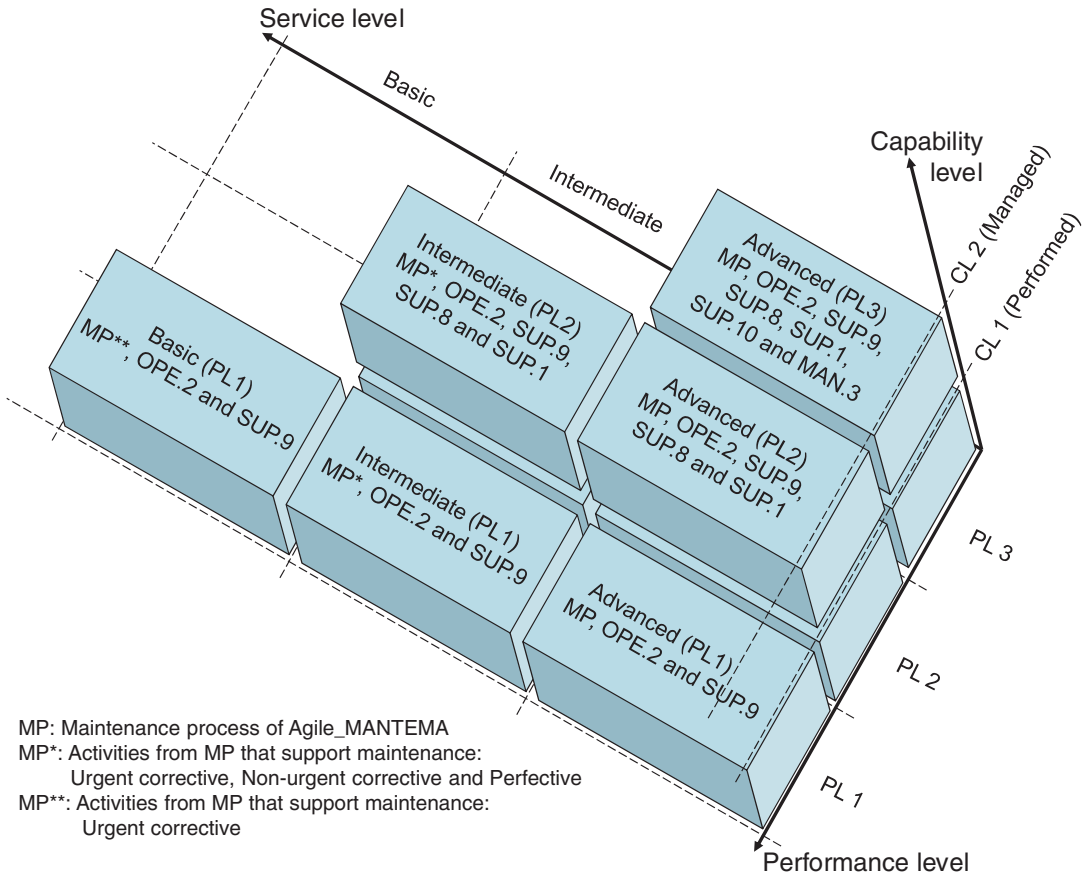

Figure 4. Three-dimensional representation of Agile_MANTEMA.

- 'Advanced' has three different performance levels (PL1, PL2 and PL3), which means that the activities described in the following processes should be carried out (i) the Agile_MANTEMA maintenance process and (ii) the auxiliary processes related with this service level. Taking the auxiliary processes into account, this service level has (i) a performance level PL1, if only the OPE.2 and SUP.9 processes are performed, (ii) PL2 performance level if the activities of the SUP.8 and SUP.1 processes are also carried out and (iii) PL3 performance level if the auxiliary processes SUP.10 and MAN.3 are also performed.

3.3. Capability levels

The concept of capability levels may be considered to be the third dimension of the representation of Agile_MANTEMA, so that on one dimension the service levels are found, while on another we find the performance levels of the processes to do with maintenance. On the third dimension are the capability levels of these processes (see Figure 4). This dimension is given to demonstrate that another way of improving and strengthening software maintenance is by means of an increase in the capability of the processes involved throughout the maintenance.

As can be observed in the above figure, both the maintenance process described by Agile_MANTEMA and the auxiliary processes may have capability levels 1 and 2. We have limited ourselves to considering only up to level two of the measurement framework for process capability, as described by ISO/IEC 15504-2 [51] and its three process attributes (from the nine attributes given by this standard). We have done this because, according to [55], the vast majority of small organizations try to achieve level 2 of capability when they start their first improvement projects.

DOI: 10.1002/smr

All the processes can be taken to capability level 2, but, as may be deduced from Figure 4, we propose that the maintenance process (or activities which are explicit in each maintenance type) described in Agile_MANTEMA (MP or MP*, MP**), together with the auxiliary processes associated with performance level 1 (OPE.2 and SUP.1), should be enhanced to reach only capability level 1 initially. Nevertheless, all the processes associated with performance levels 2 and 3 can be taken up to capability levels 1 and 2. The goal is for the small organization to establish, in a scalable way, the software maintenance strategy proposed by Agile_MANTEMA. Putting Agile_MANTEMA into practice in this scalable manner is achieved by limiting, initially, the number of processes to be defined (performance level). A limit is also placed on the quantity of practices that have to be taken into account for the definition of each of the processes. These practices are directly related to the capability level which is to be reached.

The capability level of the processes described in Figure 4 is determined by following the strategy offered by the ISO/IEC 15504-2 international standard [52]:

- Capability level 1 is characterized by its focus on the base practices of a specific process that has been established and defined by a process reference model (for instance, ISO/IEC 15504-5). In this respect, reaching level 1 is demonstrated by fulfilling the base practices and the work products described by the process which is being implemented.
- Capability level 2 is made of a set of process attributes that work together to provide a major enhancement in the capability of performing a process. These attributes are PA 2.1 Performance management and PA 2.2 Work product management. This capability level is reached by the fulfilling of the corresponding process attributes associated with this capability level. The process attributes describe capability characteristics of a process and are applicable to all processes. Process attributes are made of management practices and generic work products. A detailed description of the management practices and generic work products of the attributes PA 2.1 and PA2.2 is presented in the standard ISO/IEC 15504, Part 2 [52] and Part 5 [51].

Taking into account the three dimensions present in Agile_MANTEMA (service level, performance level and capability level), there may be different combinations for the software maintenance implementation on the part of a small enterprise. There are 9 combinations, which are related to each of the boxes set out in Figure 4. For example, we can have a maintenance process with an intermediate service level, performance level 2 and capability level 1, which indicates that the maintenance types that are carried out are urgent corrective, non-urgent and perfective. In addition to this, the base practices described in the auxiliary processes: OPE.2 Customer support, SUP.9 Problem resolution management, SUP.8 Configuration management and SUP.1 Quality assurance are largely implemented. For this maintenance process to reach capability level 2, these four processes should fully incorporate the management practices and generic work products set forth in the process attributes PA 2.1 Performance management and PA 2.2 Work product management.

The particular types of maintenance, the service levels, the performance level and the capability level will be decided according to the needs and characteristics of each organization. We also bear in mind the particular circumstances of each project or the specific maintenance service, always respecting the structure established by Agile_MANTEMA.

4. THE MAINTENANCE PROCESS

In this section, a general description of the maintenance process is given and, by way of example, a specific description of an activity and an interface with an auxiliary process is presented. A complete description of the Agile_MANTEMA and its maintenance process can be found in [56].

4.1. Description

The maintenance process aims to establish a specific guide for carrying out the requested modifications in a software product, setting out in detail what should be done, when, how and by

DOI: 10.1002/smr

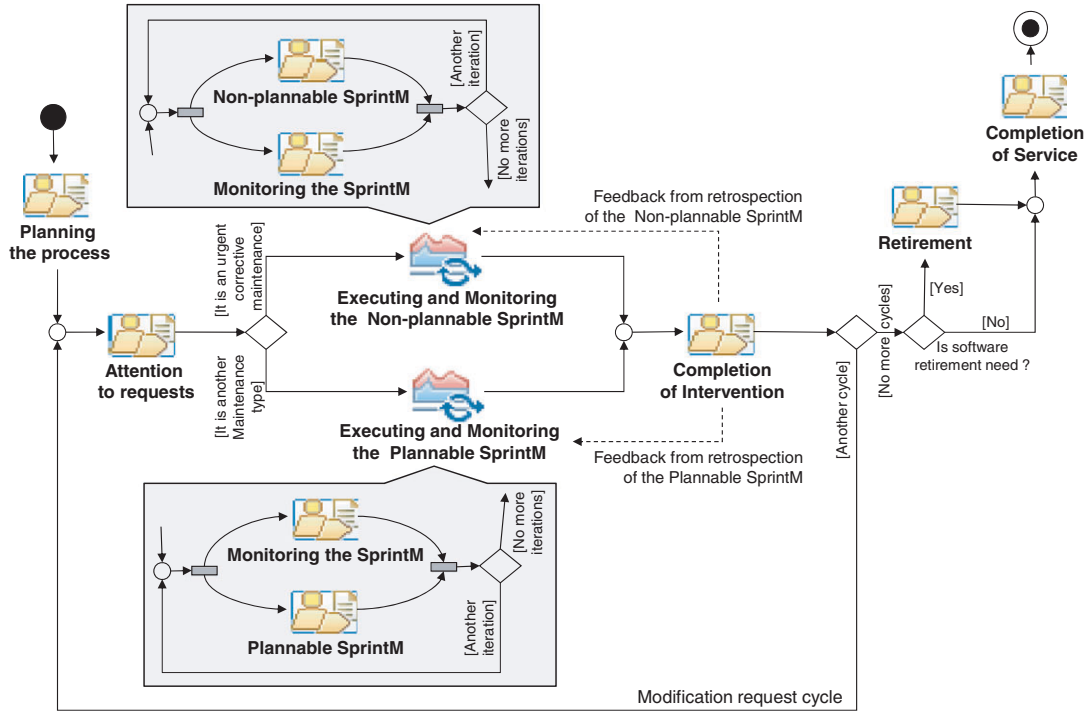

Figure 5. Flow of activities for the maintenance process in Agile_MANTEMA.

whom. Figure 5, which uses SPEM notation, displays the activities and workflow of the maintenance process defined by Agile_MANTEMA. We can see that Agile_MANTEMA establishes the following activities for carrying out the software maintenance: Planning the process, Attention to modification requests, Non-plannable SprintM, Plannable SprintM, Monitoring the SprintM, Completion of the Intervention, Software Retirement and Completion of Service. Each of these activities is described in greater detail by means of a breakdown into activities and tasks. Each task specifies in explicit terms the inputs, outputs, staff member/s in charge, techniques which may be used, along with the interfaces with the auxiliary processes that support each task.

At this point we give a very summarized description of the maintenance process defined by Agile_MANTEMA:

- It begins when the 'Maintainer' and the 'Client' begin to work together, i.e., they assign the people responsible, together with the criteria and explanation of how the work is going to be conducted. These tasks are grouped into the activity called *Planning the process*.
- When the above activity has concluded, the cycle to be followed by each 'Modification Request' received is started. The first activity to be performed in this cycle is *Attention to modification request*, by means of which a 'Modification Request' is formulated or received. This then passes into the hands of the 'Request Manager', who will be responsible for assigning the type and priority of the modification request. The group of requests, arranged in order, is called the 'Request Register'. From this register, a first group of requests, called 'Request Waiting List', is selected. This group can enter two different types of Maintenance Sprint (SprintM)[‡]: a short one for non-plannable requests and another longer one for the plannable type. Within SprintM a series of meetings will be held, to obtain the state of progress and

[‡]SprintM: Basic maintenance cycle with a duration that is recommended according to the type of maintenance. For urgent corrective from one to seven days, for the others from eight to fifteen days. In that time a modification request is dealt with and resolved. (Definition created for software maintenance based on the definition of Sprint in SCRUM.)

DOI: 10.1002/smr

the possible problems that might occur in its execution. These meetings and tasks related to the SprintM management are grouped in the *Monitoring the SprintM* activity. When the list of requests chosen in a SprintM has been dealt with, the cycle is closed with the *Completion of intervention* activity. This activity has the goal of validating and verifying the product on the client's part, going on to the production, register of documents and SprintM retrospective meetings.

- To complete the maintenance process, when no more requests are going to be received because the time allowed for the project/service has expired, a final activity, known as *Completion of service*, is performed. This ends the activities of the 'Maintainer' in such a way as to create no negative repercussions for the client organization. On occasions, the *Retirement* activity may have to be conducted before performing this task, in order to apply the plan for software retirement.

The activities of *Planning the process, Retirement and Completion of Service* take place just once and do not hinder the agility of the process. It should also be said that there are the two types of SprintM, each one having a repetitive set of activities. In line with the guidelines from Scrum, a mechanism for incorporating changes exists, thanks to the fact that rapid feedback to the client is provided, as well as a fast and regular delivery of attention to requests for modification.

The organizations and roles involved in the maintenance process are:

- Client: This is the organization which owns the software receiving the maintenance service.
  - Product owner: Representing all those who have an interest in the final product. It is responsible for the following: financing the project, return on the investment in it and launching the project. As a general rule, the owner of the product formulates modification requests of a perfective or adaptive type.
- User: The user of the software. They put forward corrective modification requests (urgent and non-urgent), as well as perfective requests.
- Maintainer: They carry out the modification of the software.
  - Request Manager: It is he or she who accepts or rejects the modification requests and who decides on the type of maintenance that corresponds to each particular case. In the case of perfective modification, they keep the owner of the product (the client) abreast of the maintenance, so that its viability can be assessed. In any other type of modification request, the request is put on the 'Request Waiting List' and it is classified as regards priority.
  - (Person) Responsible for Maintenance: He or she prepares the maintenance process and establishes the norms and procedures required for the application of the methodology. This person is also in charge of carrying out the practices, values and rules of Scrum. He or she is, moreover, a member of the maintenance team and works alongside the rest of its components, coordinating the fixed meetings of that team and ensuring that any and every possible obstacle is removed.
  - Maintenance team: This is the group of people who implement the modification requests. The team has the authority to reorganize and define the actions needed or to suggest the removal of any hindrances.

This maintenance process proposed by Agile_MANTEMA has been modelled with SPEM 2.0 [57] using the EPF Composer editor [58] (see Figure 6). Employing this tool, the activities, tasks, roles, work products and guidelines needed to carry out the software maintenance by small companies are described and are made available on the WEB.

4.2. Example of an activity

By way of example, the sub-activities and tasks of the *non-plannable SprintM* activity are set out in the following lines, giving details of the inputs, outputs, staff responsible and the techniques in each task. The goal of *non-plannable SprintM* is to offer urgent attention to the modification

DOI: 10.1002/smr

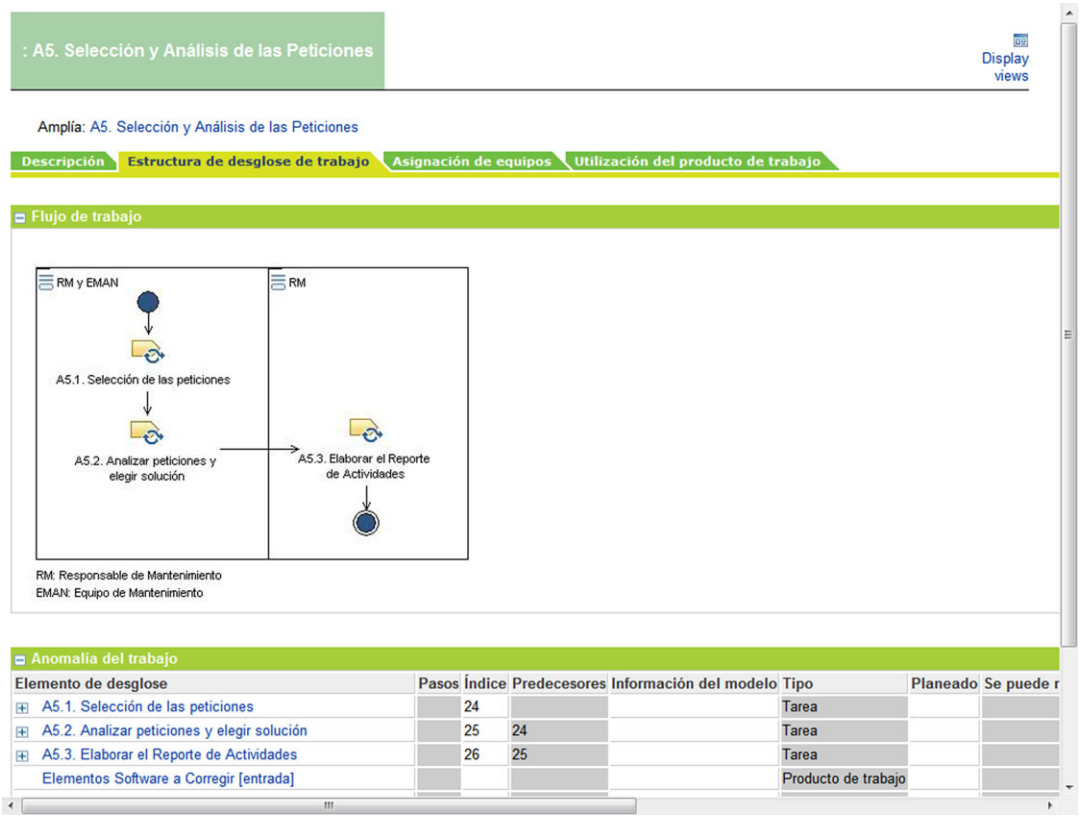

Figure 6. Screen of the Spanish version of Agile_MANTEMA implemented with EPF composer.

requests which block or interrupt the working of the product software. In other words, SprintM is run when an urgent corrective maintenance is assumed. This activity is performed when the error[§] outlined in the request for modification seriously paralyses the normal working of the system or of the organization, making correction of the error an immediate priority. The execution of short non-plannable SprintMs of between one and seven days (depending on the type of error) is recommended, with management meetings every day.

Figure 7 shows the activity diagram of the *non-plannable SprintM* and the relationships between the tasks of this activity and the tasks of the *Monitoring the SprintM* activity. Both activities are integrated, to guarantee the strategy of agile project management proposed by Scrum. Thus the *non-plannable SprintM* activity is made of the following two sub-activities:

- The Sub-Activity SNP1. Error analysis. It is composed of one single task:
  - *Task SNP1.1.* Investigate and Analyse Causes. The 'Maintenance team' analyses the 'Modification request', and checks out the problem with the help of the 'User' who made the request; it reproduces the error. It also looks at different alternatives for the implementation of the modification to correct the error and then a list of the software elements to be corrected (models, routines, documents, etc.) is drawn up.
- Sub-Activity SNP2. Urgent corrective intervention. The activity is made of two tasks:
  - *Task SNP2.1.* Performing corrective actions. The 'Maintenance team' performs the actions needed to correct the problem detected. All the components of the software product (routines, data bases etc.), that are affected by the intervention must be identified.

[§]Failure (problem or defect) in the software product, found in the executable code.

DOI: 10.1002/smr

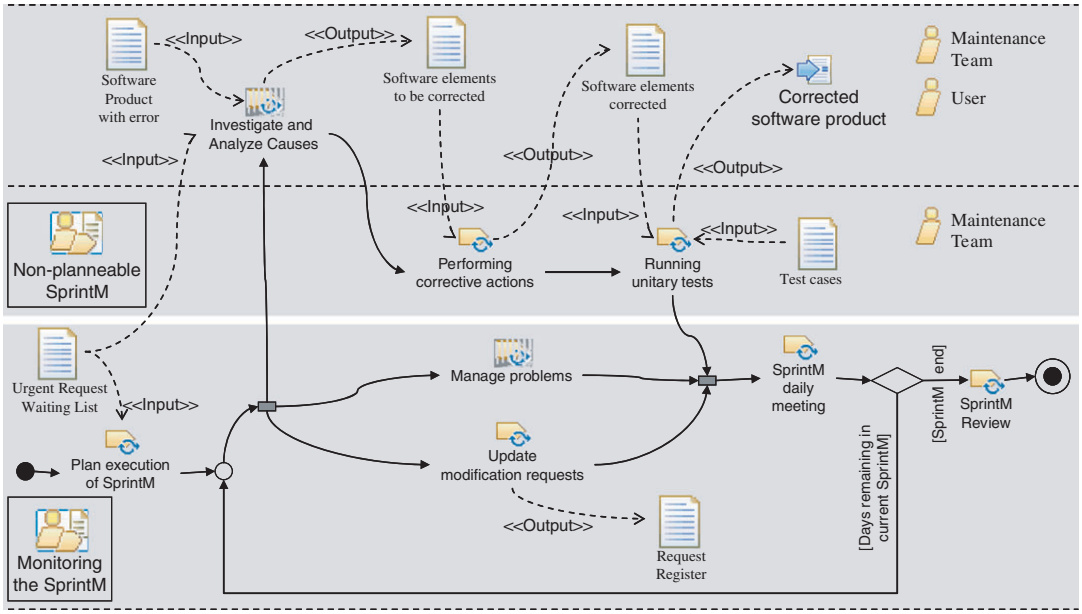

Figure 7. Activities diagram of Non-plannable SprintM and its relationship with Monitoring the SprintM.

Table IV. Example of task description.

	Inputs	Outputs	Techniques	Roles	Interfaces with other processes	Service level
Task SNP1.1 Investigate and Analyse Causes	• Software product being used, with critical error • Modification Request	Set of Software elements to be corrected	• Study of Documentation • Investigate the Software Product • Observation and interviews	• Maintenance Team • User	OPE.2 Customer support	Basic

- *Task SNP2.2.* Running unitary tests. The 'Maintenance team' should check the proper functioning of all the changes made. The tests conducted must be documented in the 'Unitary Tests Carried Out' document. This task does the job of checking on the correct operation of the module on which the corrective actions have been carried out.

The tasks included in Agile_MANTEMA are outlined by means of the corresponding input products, output products, roles involved and techniques (which can be used to guide the task). In addition, the relationship of each task with other processes and the service levels which that task supports are also described (see Table IV).

In an urgent corrective maintenance (which involves a critical error), the priority is to have the software product working as soon as possible. This does not allow tasks to be carried out in a systematic and formal way. However, to address an error of this type, its causes need to be investigated and analysed. This task is often performed, however, under so much pressure that there is an ensuing loss of valuable information which might have been very useful for addressing the present and future problems. One of our purposes in defining this task is that the maintenance team: (i) acquires awareness when it is carried out, and (ii) stores the sensitive information about the error and the problem. This can allow the organization to have a better way of managing and resolving this and other maintenance types.

DOI: 10.1002/smr

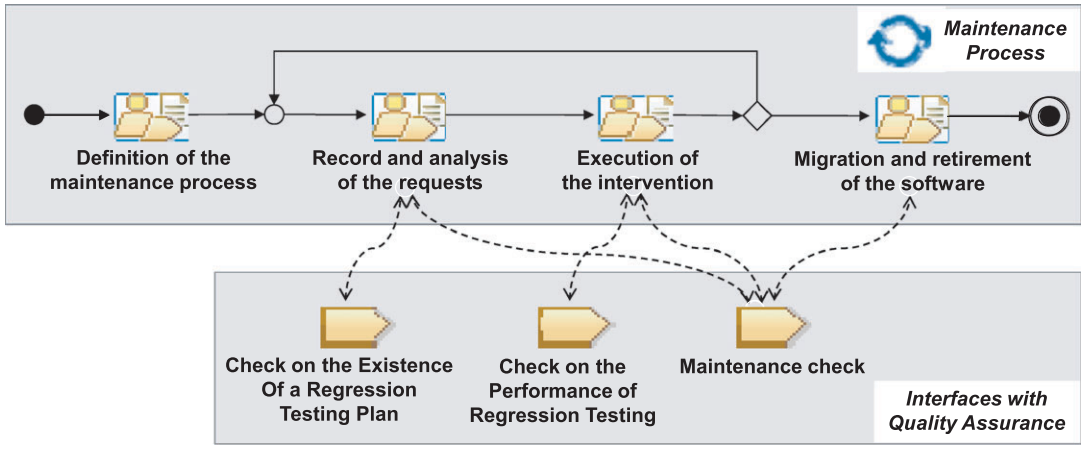

Figure 8. Interface of the maintenance process with Quality Assurance.

4.3. Example of an interface with an auxiliary process

This section gives an example of the interface of the Quality Assurance process with which the maintenance process is involved in the intermediate and advanced service levels (see Table II). With this interface the person 'Responsible for Maintenance' carries out periodic check-ups on the maintenance process, aiming to verify that the modification requests that have been made are being put into operation properly.

To carry out the modification request, it may be necessary to check on some elements described by the maintenance process:

- The content of the plan for regression testing.
- The running of the regression testing, according to the strategy laid out in the Quality Assurance plan.
- The verifications and test cases that have been included in the test plan, which are related to the changes produced by a modification request.
- The problems that have arisen and that have been detected during the running of the maintenance process, aiming to find out if some quality property of the product being maintained could be affected.

In Figure 8 the tasks (interfaces) of Quality Assurance which support the maintenance process are shown and a description of these is given in the following lines:

- *Task QA1: Maintenance check.* A strategy for Quality Assurance of the activities and work products involved in the maintenance process should be developed. A periodic check of the *Request Register* takes place, to make sure that it is kept up-to-date. Similarly, there is a check to ensure that the user accepts or rejects the solution proposed in answer to the modification request and that he/she gives their formal approval to the closure of the request. It is also verified that products, processes and activities carried out during maintenance are in accordance with those described by Agile_MANTEMA.
- *Task QA2: Check on the Existence of a Regression Testing Plan.* A monitoring task takes place to ensure that a regression testing plan has been set up, according to the criteria established in the Quality Assurance plan. The goal is to determine what techniques are going to be applied for the execution of the tests, what the acceptance criteria will be, how the verification activities will be carried out as well as how the results will be issued. If changes are brought into the execution of the regression testing, these must be approved by the person 'Responsible for maintenance'. A check takes place on the existence or not of a strategy for the management of the test results.

DOI: 10.1002/smr

- *Task QA3: Check on the Performance of Regression Testing*. A check is performed to see whether the established regression testing, verifications and test cases have been carried out, to determine whether the changes have been introduced properly. In doing this monitoring, the strategy established for the management of the results of all these tests should be borne in mind. Wherever there are additional test cases that have been brought in as a consequence of the corrective measures taken to solve modification requests, the individual 'Responsible for maintenance' will check to see that they have been sorted out correctly. In the same way, the unresolved problems that have arisen will be reviewed, to assess to what extent the quality properties may have been compromised. The approval on the part of the person 'Responsible for maintenance' is recorded.

5. EXPERIENCE REPORT

The description of the experience report on the application of the Agile_MATEMA has been organized taking into account the indications for experience report papers recommended by [59]. In this respect, the following subsection describes this experience report in terms of: the application context, the description of organizations participating, the report and analysis of the use of the methodology proposed by companies, along with an assessment of Agile_MANTEMA, as given in the views of companies and researchers.

5.1. Application context

Agile_MANTEMA has been applied in two software firms from Uruguay (*critical reference group*) by means of the execution of the corresponding problem-solving cycle from action-research. Agile_MANTEMA has been used as a reference model in two improvement cycles, carried out in the context of the COMPETISOFT project, whose objective was to improve the software maintenance process in these two companies. These improvement cycles were carried out over a period of six months and in action-research terms they were supported by the Software Engineering Group in the Engineering Faculty of the University of the Republic of Uruguay (*advisers*). The cycles were directed using as their improvement guide the activities established by the improvement process defined in PmCOMPETISOFT [60]. They also used the proposal of Agile_MANTEMA (as reference for best practice in software maintenance), aiming to direct the maintenance process in each of the enterprises.

One of the activities established by PmCOMPETISOFT is the performance of *Diagnosing the Process* in relation to the initial situation of the enterprise, contrasting the process that the firm carries out against a reference model (in this case the maintenance process of Agile_MANTEMA). This diagnosis allowed relevant information to be obtained about the general state of the maintenance process carried out by each of the firms. That information was analysed, thereby generating the results of the diagnosis, aiming to discover the improvement opportunities in the maintenance process (the improvement opportunities are best practices that the company is not undertaking in its process). These good practices were incorporated into the firms' maintenance process by means of the activities of *Formulating and Executing Improvement* from PmCOMPETISOFT.

5.2. Characteristics of the firms

At this point, we will focus on an introduction to the characteristics of the firms involved in the application of Agile_MANTEMA. In an effort to respect the need for confidentiality of the enterprises involved in the application of Agile_MANTEMA, we call these firms EU1 and EU2. Their main features are:

- EU1 has been in the market for 18 years and has 45 employees at present. It has a single software product, which has been developed by the company itself. The product is highly adaptable to the characteristics of different organizations and the firm has been very successful in its commercialization, so at the moment there is a very broad business base established

DOI: 10.1002/smr

for it. The adaptability of the product has made it possible to introduce it into different organizational settings, with no need to modify the software product. Yet, many clients ask for modifications or special additions to it. In EU1, the process used for the maintenance of the single product of the firm was analysed.

- EU2 has had a position in the market place for 15 years and currently has 26 employees. It has different software systems, some made-to-measure for a particular client. Other more generic products were put into operation with a limited number of clients. In general, the firm's customers are large organizations and each installation of the products has a considerable number of users. In EU2, there was an analysis of the maintenance process used in what the company believed to be its best-organized and best-working project. It is a system that has been developed for a large organization (employed by dozens of users) and the service quality that the organization renders to thousands of customers depends on its operating properly.

5.3. Reporting and analysing the work performed in companies

In this subsection, we present a general description of the use of Agile_MANTEMA by both companies, based on contrasting the companies' maintenance process with the proposed maintenance strategy.

As regards the types of Maintenance defined by Agile_MANTEMA, both firms claim that the advanced service level is offered, which includes: Urgent Corrective, Non-Urgent Corrective, Perfective, Adaptive and Preventative. Furthermore, in both firms there was an initial analysis of the existing roles assigned to maintenance tasks and their correspondence to the roles defined by Agile_MANTEMA.

- In EU1, the role of Functional Analyst exists. This person acts as a representative of the client and takes on, in part, the role of 'Product Owner' defined in Agile_MANTEMA, especially in those aspects of the product that are applicable to the totality of customers and users. There is, apart from this, a person who fulfils the roles of person 'Responsible for maintenance' and 'Request Manager'. A 'Maintenance team' is also identified.
- On the other hand, in EU2 the 'Product Owner' is strictly on the client's side. The role of Functional Analyst also exists, who is at the same time 'Request Manager' and person 'Responsible for maintenance'.

On the other hand, contrasting the activities established by Agile_MANTEMA for software maintenance with those carried out in the two firms, the improvement opportunities were found and these are presented below. These improvement opportunities were incorporated into the maintenance processes of the firms by means of the improvement cycle that each company carried out. This improvement cycle followed the activities described by PmCOMPETISOFT [60].

- Formal definition of the maintenance process in both organizations. This process was improved by using practices described in Agile_MANTEMA, since in both firms some of the activities established by the proposed methodology were being carried out, but the maintenance process was neither complete nor formally defined.
- Definition of a strategy for management of modification requests for EU1. In this firm, before facing the maintenance improvement cycle, a development process improvement cycle was undertaken. By means of this, the documentation of the system requirements and the specification of the requirements applicable to each modification request were systematized. The improvement of the maintenance process allowed us to define the way of up-dating the requirements specifications which are applicable to modified software products, essentially by means of the generation of new versions of the documents involved. At the moment, the issue of handling the up-dating of the system requirement is being resolved, especially for the case of those requirements which may have changed. This is because generating a new version for each SprintM brings about a very heavy workload, which is difficult for the firm to take on. One alternative which has been assessed is to keep the original version of the requirements document and complement it with some 'deltas' which correspond to the requests dealt with.

DOI: 10.1002/smr

- Development of a new system for the requests management in EU2, due to the fact that there are various different channels of communication (telephone, e-mail, verbal communication) in this firm. Moreover, there are no guarantees that all the requests will be dealt with. This system was developed during the improvement cycle of the maintenance process and it is due to go into production soon. It has two goals: (i) to become the only way of communication related to modification requests, and (ii) the user can register and see the state of the requests through a Web interface that is friendly and easy to use.
- Incorporation of a document for the traceability of the modifications carried out during a maintenance intervention. The recording of the intervention is carried out in EU1 as the treatment of a request progresses, using the system of request management. To improve this, we saw the advisability of incorporating a document in which the changes carried out effectively were registered, identifying the components involved. For the case of corrective maintenance, this allows us to identify modules liable to be corrected. In the final register of corrective maintenance the phase (requirements, design, construction, among others) in which the error came is indicated. The goal is to guide the improvement of the development process towards the phase which is the main source of the problems.
- Drawing up of a document for the specification of modifications and incorporation of the activity Monitoring the SprintM by the 'Maintenance team' of the EU2 firm in an effort to assess how well the maintenance sprint was carried out. In both companies, a follow-up check did take place, but not in the form of 'Regular Meetings' based on Scrum, as Agile_MANTEMA proposes. In EU2, this practice was introduced so that the whole team would be working together and in the same direction, making the most of their own abilities to remove obstacles that its members found when carrying out their tasks.
- Incorporation by enterprise EU2 of the Review and Retrospective Meeting proposed by Agile_MANTEMA. Neither of the firms had been performing this activity.

5.4. Discussion

Both the firms who took part and used Agile_MANTEMA expressed the views that this methodology: (i) was an important and practical aid in making them reflect on their maintenance process, and (ii) was useful in tackling improvement in this process. The researchers observed that the proposal put forward by Agile_MANTEMA turns out to be suitable as a reference model for the improvement of the maintenance process in small firms. In addition, the Scrum approach is consistent with the way the maintenance staff of these enterprises see the maintenance process. From these experiences in real settings, it can be highlighted that the use of Agile_MANTEMA was enriching, both for the improvement group in charge of the application and for the research group which created the methodology.

We will now go on to highlight some aspects of the application of Agile_MANTEMA in small organizations, first in terms of the lessons learned from views of the companies and researchers, and then in terms of the limitations of the results and conclusions.

5.4.1. Lessons learned. Some relevant lessons learned which can be taken from the two use cases are:

- Among the issues that most concerned the person 'Responsible for maintenance' in EU1 was to be able to establish and then manage the commitments taken on by the 'Maintenance team'. These consisted of dealing with the demands of their various clients and of letting them know in advance if it was going to turn out to be impossible to fulfil the agreed timescale. The adviser showed the person 'Responsible for maintenance' that this matter is related to the maintenance project management. The adviser also pointed out that Agile_MANTEMA proposes that this topic should be tackled by: the agile management of the maintenance process, by means of the Monitoring the SprintM activity and (ii) the interface with the auxiliary process MAN.3 Project management, defined for the advanced service level. The person 'Responsible for maintenance' opted to bring in the activity of Monitoring the SprintM initially, but expressed

DOI: 10.1002/smr

interest in incorporating the tasks associated with the project management interface in the improvement effort that followed.

- On analysing the modification requests scenario in EU1, the person 'Responsible for maintenance' detected that there were some requests still waiting to be dealt with from over three years before. It was thus clearly advisable to introduce the consideration of new elements related to requests analysis in Agile_MANTEMA. Among other aspects, the work load needed to be analysed, along with its evolution, the concentration of requests in certain components (especially in the case of corrective maintenance), priorities, the validity of the requests sent in a long time beforehand and the requests that may be inter-related. These aspects have to do with the management of projects and/or products. With regard to project management, the strategy outlined under the previous bullet point can be used. As far as the management of the 'Request Register', which is a work product, is concerned, this issue can be dealt with by integrating the management practices described by the process attributes 2.2 Work product management. In other words, the maintenance process described by Agile_MANTEMA must be increased to capability level 2.
- Those 'Responsible for maintenance' in both firms believe that the general approach of Scrum included in Agile_MANTEMA allows us to combine conflicting objectives, given the emphasis that it places on short-term planning according to the priorities of the Modification Requests existing at any given moment of time. These would include goals such as the quest for efficiency in execution and the desire to give sufficient attention to changes in the priorities of these requests.
- Both the companies considered that the grouping of requests for short-term planning, as presented in Plannable SprintM, offers benefits. These include: (i) allowing us to assess the compatibility and consistency of the changes associated with a group of requests, and (ii) permitting improvement in the efficiency of the execution, since it is possible to perform one single instance of regression testing for the group of requests.
- On the basis of an analysis of the maintenance processes in both firms, it is clear that the emergency repairs (requests of urgent corrective maintenance) should afterwards be directed towards a Plannable SprintM. The goals are to guarantee that the interventions carried out maintain the required level of quality, that they complete the tests that may not have been carried out and that they ensure that the problem which prompted the urgent request is solved in subsequent versions of the software.
- Software maintenance management involves different information which comes from various sources. Consequently, being able to count on a System of Request Management which allows us to handle and manage the documentation of the interventions is of great assistance when putting changes into effect during the maintenance process.

Among the most relevant aspects to highlight in the work carried out in each of the companies, we draw attention to the following:

- In EU1, when the maintenance process underwent a diagnosis with Agile_MANTEMA, they found that it was possible to recognize how a large part of the practices and activities undertaken in the firm fitted in somewhere in Agile_MANTEMA. Nevertheless, it was clear that the maintenance process being followed by the company lacked various best practices that the maintenance methodology suggested (in other words, these practices were not being carried out by the firm). To improve the maintenance process, four inter-related aspects were worked on: (i) defining roles and responsibilities, (ii) improving mechanisms for communication and documentation, (iii) systematizing the documentation of the interventions and (iv) adjusting the request management system.
- In EU2, when the maintenance process of the Project under analysis underwent diagnosis with Agile_MANTEMA, they discovered that they were already grouping requests in accordance with a SprintM. It should also be noted that, as in the case of EU1, they detected that their greatest improvement opportunities had to do with the handling and management of maintenance requests and with subsequent intervention. That being so, the improvement activities were directed towards strengthening those aspects. Thus, this firm, which did not

DOI: 10.1002/smr

have a system for the management of requests, opted to create one, so that clients and users could register their requests and also consult to see what stage these were at. The idea was to have a channel for all communication with the 'Maintenance team'. Apart from that aspect, they also brought in agile follow-up practices to the maintenance process activities.

Furthermore, alongside the learning and reflection on the part of the enterprises, the researcher group in charge of the creation of the maintenance methodology also obtained an initial assessment of its use in a real environment. With the feedback given by the employees of the firms who used the maintenance methodology, valuable information for carrying out some subsequent improvements in it was also gathered. Such improvements have focused on refining Agile_MANTEMA, with the following suggestions:

- Provide support tasks to the communication between the 'Maintenance team' and the user, since this is a decisive aspect of the maintenance process.
- Incorporate in a SprintM the whole set of tasks for the verification and validation of the consistency of the set of 'Modification Requests' involved.
- Substitute the task 'Execute the software simultaneously' in a Plannable SprintM with 'Regression Testing', which is more general and more suitable.
- Indicate explicitly that an urgent request which starts off a Non-Plannable SprintM can receive unitary attention, that is to say it does not have to be grouped with any other request.
- Include in the register of the activity 'Complete Intervention' of the corrective maintenance, an indication of the specific phase (analysis, design, construction, test) in which the error occurred.
- Take the task 'Pass to production' out of the activity 'Complete Intervention', since it is common to find that the release and deployment by the client of a modified software product is an independent activity, separate from the completion of the intervention.
- Create predefined templates which are applicable to the different activities established by the methodology, to make it easier to put Agile_MANTEMA into operation.

At the moment, the lessons learned and the suggestions for improvement in the methodology are being analysed, the goal being to integrate them into a new version of Agile_MANTEMA, through the carrying out of a new research cycle of action-research.

From the experience reports shared by the companies, we have obtained relevant information about the application of Agile_MANTEMA in the industrial setting. Based on this information, we consider that this maintenance methodology can be useful and practical for supporting the work related to software maintenance in small organizations. We say this because both companies have reported positive experiences and lessons on the usefulness of such methodology in improving their maintenance process. Among other aspects, these experiences are related to a better knowledge of its software maintenance process, the incorporation and implementation of new best practices (technical and management) in this process and the establishing of the type of maintenance that they offer. In short, the work carried out and reported by these companies indicates that they have reflected on the way they perform their maintenance process and that they have improved their software maintenance process by using the strategy proposed by Agile_MANTEMA.

5.4.2. Limitations The participative variant of Action-research was the research strategy used to carry out the application of Agile_MANTEMA in the two companies that are part of the critical reference group, and they had the support of an adviser. This person participated in the diagnosis of the maintenance process of each company and gave recommendations to them as to how to improve their respective processes. These companies put in place the recommendations made by the adviser and they informed him of the results and effects through experience reports. It is important to highlight that the adviser did not participate in the incorporation of the improvement opportunities of the maintenance process (that is to say, in the organizational change). This person therefore had a limited control of the work performed on this topic within the

DOI: 10.1002/smr

companies. This aspect entailed an application of Agile_MATEMA which was not very systematic or controlled, from the point of view of the researchers. That limits the results and conclusions obtained.

Furthermore, the study carried out on the application of Agile_MANTEMA also presents other limitations: (i) the observations and conclusions presented are based on two experience reports from two companies, which can limit the power of generalization, and (ii) there may be a bias in the experience reports, due to some particular kind of handling of events and information by companies and the adviser, since there is only one evidence source for the application, which is the experience report shared by the company.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented Agile_MANTEMA, which puts forward a software maintenance methodology that focuses on small organizations. This methodology has been developed in the context of the COMPETISOFT project, where the enterprises and some researchers participating in the project highlighted the need for a strategy that would specifically address the activities related to software maintenance, separating these from the ones done for software development. In this respect, Agile_MANTEMA sets out the elements needed to guide the software maintenance in small organizations step-by-step. Furthermore, this paper has also described the experience report of two small firms who carried out an improvement cycle which focused on the maintenance process, doing so by following the maintenance strategy described by this methodology. From this initial employment of Agile_MANTEMA in these two small companies, it was seen that the maintenance methodology proposed can be useful and suitable for this type of company, since both firms have formally defined their maintenance process based on Agile_MANTEMA, thus attempting to improve their daily routine as regards software maintenance.

For the construction of the components of Agile_MANTEMA, *process/method researchers* have carried out several research cycles (of different types). The first research cycles were of a conceptual type, to get to know the MANTEMA methodology, along with the Scrum method. The idea was to see how to use them in an integrated way to construct Agile_MANTEMA. Then various methodological research cycles were executed, to develop the different components of the proposed methodology (maintenance process, services levels, performance levels and capability levels). Afterwards, by means of the execution of a new technical research cycle, an electronic process guide was also constructed (by modelling Agile-MANTEMA with SPEM and then publishing it with the EPF composer). Finally, the released version of Agile_MANTEMA was applied in two companies (of the *critical reference group* of COMPETISOFT) through a problem-solving cycle, and the feedback of this application was carried out with experience reports from companies. The application did not follow a method that would guarantee a controlled environment by researchers. A real work was used instead, consisting of the collaboration of company professionals to assess the proposed methodology in a practical way. Although experience reports are not the most suitable way of validating a research proposal, this method allows us to obtain results regarding the application of software technologies, as well as their effectiveness in industrial settings [61]. This is useful for researchers in identifying issues related to the research. It is also an asset for organizations if we find out about the use of these technologies in a real context. In this respect, the experience report described in this paper has been useful as an initial empirical validation of Agile_MANTEMA. On the basis of what it tells us, we believe that it is advisable and even necessary to use a more comprehensive empirical method (for instance case study) to validate this methodology in future applications of it. This new method should allow us to strengthen and increase the rigour of the application of the methodology proposed and to address the limitations described by means of a validation plan. The objective would be to increase the reliability of the results obtained and the conclusions drawn.

In the future work, the suggestions for improvement registered in the experience reports will be incorporated into a new version of Agile_MANTEMA. Furthermore, we have come to the

DOI: 10.1002/smr

conclusion that we have to strengthen our maintenance approach. To do so, we will take into account different elements from the corrective maintenance maturity model (CM3). This methodology will, moreover, be applied in other small enterprises which may be interested in its use when employing an empirical research method as a case study. The objective is that from a representative set of case studies new feedback may be obtained, so that we may go on to carry out further assessment, fine-tuning, improvement and validation of the maintenance methodology we have proposed.

ACKNOWLEDGEMENTS

We acknowledge the assistance of the following projects: ARMONIAS (PII2109-0223-7948, JCCM of Spain), PEGASO-MAGO (TIN2009-13718-C02-01, FEDER and MEC of Spain) and INGENIOSO (PEII11-0025-9533, JCCM of Spain). Acknowledgements by Francisco J. Pino to the University of Cauca where he works as Associate Professor.

REFERENCES

1. Pigoski T. Practical software maintenance. *Best Practices for Managing your Investment*. Wiley: New York, 1997.
2. Canfora G, Cimitile A. Software maintenance. In *Handbook of Software Engineering and Knowledge Engineering*, vol. 1. Fundamentals. Chang SK (ed.). World Scientific Publishing Co. Pte. Ltd.: Singapore, 2001; 91–120.
3. Kajko-Mattsson M. Corrective maintenance maturity model: Problem management (PhD Synopsis). *The 18th International Conference on Software Maintenance* (*ICSM 2002*). IEEE Computer Society: Silver Spring MD, 2002; 486–490.
4. Huang SJ, Lai R. Measuring the maintainability of a communication protocol based on its formal specification. *IEEE Transactions on Software Engineering* 2003; **29**(4):327–344.
5. O'Keeffe M, Ó Cinnéide M. Search-based refactoring for software maintenance. *Journal of Systems and Software* 2008; **81**(4):502–516.
6. Singer J. Practices of software maintenance. *Proceedings International Conference on Software Maintenance*. IEEE Computer Society Press: Los Alamitos CA, 1998; 139–245.
7. Krogstie J, Jahr A, Sjøberg DIK. A longitudinal study of development and maintenance in Norway: Report from the 2003 investigation. *Information and Software Technology* 2006; **48**(11):993–1005.
8. MBI. Panorama de la Industria del Software en Latinoamérica, 2004. Available at: http://www.mbi.com.br/MBI/biblioteca/relatorios/200409panswlatam/200409_panorama_industria_software_america_latina.pdf [April 2011].
9. April A, Huffman J, Abran A, Dumke R. Software maintenance maturity model (SMmm): The software maintenance process model. *Journal of Software Maintenance and Evolution*: *Research and Practice* 2005; **17**(3):197–223.
10. Van Bon J. *World Class IT Service Management Guide 2000*. ten Hagem & Stam Publishers: The Hague, Netherlands, 2000.
11. Batista J, Figueiredo A. SPI in a very small team: A case with CMM. *Software Process*: *Improvement and Practice* 2000; **5**(4):243–250.
12. Polo M, Piattini M, Ruiz F. Using a qualitative research method for building a software maintenance methodology. *Software Practice and Experience* 2002; **32**(13):1239–1260.
13. Zitouni M, Abran A, Bourque P. Élaboration d'un outil d'évaluation et d'amélioration du processus de la maintenance des logiciels: Une piste de recherche. *Proceedings Génie Logiciel et ses Applications*, *Huitiémes Journées Internationales*, 1995; 727–739.
14. Polo M, Piattini M, Ruiz F, Calero C. MANTEMA: A software maintenance methodology based on the ISO/IEC 12207 Standard. *Proceedings of the Fourth IEEE International Symposium and Forum on Software Engineering Standards*, 1999; 76–81.
15. Polo M, Piattini M, Ruiz F. Improving the quality of the maintenance process. *Second World Congress for Software Quality* (*WCSQ 2000*), 2000, 325–330.
16. Oktaba H, Garcia F, Piattini M, Pino F, Alquicira C, Ruiz F. Software process improvement: The COMPETISOFT project. *IEEE Computer* 2007; **40**(10):21–28.
17. Pino F, Pardo C, García F, Piattini M. Assessment methodology for software process improvement in small organizations. *Information and Software Technology* 2010; **52**(10):1044–1061.
18. Schwaber K. The Scrum development process. *OOPSLA '95 Workshop on Business Object Design and Implementation*. ACM Press: Austin TX, U.S.A., 1995; 1–23.
19. ISO. ISO/IEC 12207: Information technology—Software life cycle processes, 2002. Available at: www.iso.org [April 2011].
20. Coleman G, O'Connor R. Investigating software process in practice: A grounded theory perspective. *Journal of Systems and Software* 2008; **81**(5):772–784.
21. Hurtado J, Bastarrica C. Implementing CMMI using a combination of agile methods. *CLEI Electronic Journal* 2006; **9**:1–15.

DOI: 10.1002/smr

22. Abrahamsson P, Salo O, Rankainen J, Warsta J. *Agile Software Development Methods*: *Review and Analysis*. VTT Publications: Finland, 2002; 478.
23. Ambler S. *Agile Modeling*: *Effective Practices for Extreme Programming and the Unified Process*. Wiley: New York, 2002.
24. Beck K. Embracing change with extreme programming. *IEEE Computer* 1999; **32**:70–77.
25. Hunt A, Thomas D. *The Pragmatic Programmer*. Addison-Wesley: Reading MA, 2000.
26. Schwaber K. Scrum Guide, 2010. Available at: http://www.scrum.org/storage/scrumguides/Scrum%20Guide.pdf [April 2011].
27. Beck K. Principles behind the Agile Manifesto, 2003. Available at: http://agilemanifesto.org/principles.html [April 2011].
28. Poole C, Huisman J. Using extreme programming in a maintenance environment. *IEEE Software* 2001; **18**(6):42–50.
29. Svensson H, Höst M. Introducing an agile process in a software maintenance and evolution organization. *The Ninth European Conference on Software Maintenance and Reengineering* (*CSMR'05*). IEEE Computer Society: Silver Spring MD, 2005; 256–264.
30. Kajko-Mattsson M, Forssander S, Olsson U. Corrective maintenance maturity model (CM3): Maintainer's education and training. *The 23rd International Conference on Software Engineering* (*ICSE 2001*). IEEE Computer Society: Toronto, Canada, 2001; 610–619.
31. Kajko-Mattsson M. Problem management maturity within corrective maintenance. *Journal of Software Maintenance and Evolution*: *Research and Practice* 2002; **14**(3):197–227.
32. Kajko-Mattsson M. CM3: Problem management: Taxonomy of activities. *IASTED International Multi-conference on Applied Informatics Innsbruck*, 2003; 1161–1166.
33. Kajko-Mattsson M, Ahnlund C, Lundberg E. CM3: Service level agreement. *The 20th International Conference on Software Maintenance* (*ICSM 2004*). IEEE Computer Society: Chicago, U.S.A., 2004; 432–436.
34. Kajko-Mattsson M, Winther P, Vang B, Petersen A. An outline of CM3: Emergency problem management. *The 31st EUROMICRO Conference on Software Engineering and Advanced Applications* (*EUROMICRO-SEAA 2005*). IEEE Computer Society: Porto, Portugal, 2005; 292–303.
35. Kajko-Mattsson M. Evaluation of CM3: Front-end problem management within industry. *The 10th European Conference on Software Maintenance and Reengineering* (*CSMR 2006*). IEEE Computer Society: Silver Spring MD, 2006; 367–368.
36. Calvo-Manzano JA, Cuevas G, San Feliu T, De Amescua A, Pérez M. Experiences in the application of software process improvement in SMES. *Software Quality Journal* 2002; **10**(3):261–273.
37. Scott L, Jeffery R, Carvalho L, D'Ambra J, Rutherford P. Practical software process improvement—The IMPACT project. *Proceedings of the Australian Software Engineering Conference*, 2001; 182–189.
38. Wangenheim CGV, Weber S, Rossa Hauck JC, Trentin G. Experiences on establishing software processes in small companies. *Information and Software Technology* 2006; **48**(9):890–900.
39. Oktaba H. MoProSoft®: A Software Process Model for Small Enterprises. *Proceedings of the First International Research Workshop for Process Improvement in Small Settings*, Pittsburgh, Carnegie Mellon University, 2006; 93–101.
40. Weber K, Araújo E, Rocha A, Machado C, Scalet D, Salviano C. Brazilian software process reference model and assessment method. *Computer and Information Sciences* (*Lecture Notes in Computer Science*, vol. 3733). Springer: Berlin/Heidelberg, 2005; 402–411.
41. McCaffery F, Richardson I, Coleman G. Adept—A software process appraisal method for small to medium-sized Irish software development organisations. *European Systems & Software Process Improvement and Innovation* (*EuroSPI 2006*), 2006; 7.12–7.21.
42. Cater-Steel AP, Toleman M, Rout T. Process improvement for small firms: An evaluation of the RAPID assessment-based method. *Information and Software Technology* 2006; **48**(5):323–334.
43. Horvat RV, Rozman I, Györkös J. Managing the complexity of SPI in small companies. *Software Process*: *Improvement and Practice* 2000; **5**(1):45–54.
44. McKay J, Marshall P. The dual imperatives of action research. *Information Technology and People* (*Special Issue on Action Research in Information Systems*) 2001; **14**(1):46–59.
45. Chiasson M, Germonprez M, Mathiassen L. Pluralist action research: A review of the information systems literature. *Information Systems Journal* 2009; **19**(1):31–54.
46. Avison D, Lan F, Myers M, Nielsen A. Action research. *Communications of the ACM* 1999; **42**(1):94–97.
47. Wadsworth Y. What is participatory action research? Action Research International (Paper 2), 1998.
48. French WL, Bell CH. *Organization Development*: *Behavioral Science Interventions for Organization Improvement*. Prentice-Hall: London, 1999.
49. ISO. ISO/IEC 12207:Systems and software engineering—Software life cycle processes, 2006. Available at: www.iso.org [April 2011].
50. MAP. Métrica Versión 3. Metodología de Planificación, Desarrollo y Mantenimiento de sistemas de información, 2007. Available at: http://www.csi.map.es/csi/metrica3/index.html [April 2011].
51. ISO. Information technology—Process assessment—Part 5: An exemplar process assessment model. ISO/IEC 15504-5:2006(E), 2006. Available at: www.iso.org [April 2011].
52. ISO. Information technology—Process assessment—Part 2: Performing an assessment. ISO/IEC 15504-2:2003/Cor.1:2004(E), 2004. Available at: www.iso.org [April 2011].

DOI: 10.1002/smr

53. Ruiz F, Piattini M, Polo M, Calero C. Maintenance types in the MANTEMA Methodology. *International Conference on Enterprise Information System*, 1999; 27–30.
54. Pino F, Garcia F, Piattini M. Key processes to start software process improvement in small companies. *The 24th Annual ACM Symposium on Applied Computing* (*SAC'09*), 2009; 509–516.
55. Pino F, Garcia F, Piattini M. Software process improvement in small and medium software enterprises: A systematic review. *Software Quality Journal* 2008; **16**(2):237–261.
56. Pino F, Ruiz F, Salas S. Agil_Mantema Versión 1.0. COMPETISOFT IT.21 (In Spanish). Available at: http://alarcos.inf-cr.uclm.es/competisoft/ [April 2011].
57. OMG. Software & Systems Process Engineering Metamodel specification (SPEM) Version 2.0, 2008. Available at: http://www.omg.org/spec/SPEM/2.0/ [April 2011].
58. Eclipse. Eclipse Process Framework Project (EPF), 2007. Available at: http://www.eclipse.org/epf/ [April 2011].
59. Montesi M, Lago P. Software engineering article types: An analysis of the literature. *Journal of Systems and Software* 2008; **81**(10):1694–1714.
60. Pino F, Hurtado J, Vidal J, García F, Piattini M. A process for driving process improvement in VSEs. *International Conference on Software Process* (*ICSP 2009*) (*Lecture Notes in Computer Science*, vol. 5543). Springer: Berlin. 2009; 342–353.
61. Briand L, Selby R. Empirical software engineering: Industrial experience reports. *Empirical Software Engineering* 2006; **11**(4):613.

AUTHORS' BIOGRAPHIES

Francisco J. Pino has a European PhD in Computer Science from the University of Castilla-La Mancha (UCLM), Spain. He is currently an associate professor at the Electronic and Telecommunications Engineering Faculty at the University of Cauca, in Popayán (Colombia). He is a member of the IDIS Research Group and his research interest is Software process improvement in small companies, Harmonization of multiple improvement technologies and Qualitative research methods for Software Engineering.

Francisco Ruiz is a PhD in Computer Science from the University of Castilla-La Mancha (UCLM) and an MSc in Chemistry–Physics from the University Complutense of Madrid. He is associate professor of the Department of Information Technologies and Systems at UCLM in Ciudad Real (Spain). He has been Dean of the Computer Science Faculty between 1993 and 2000. Previously, he was Director of the Computer Services in the mentioned university (1985–1989) and has also worked in private companies as analyst-programmer and project manager. His current research interests include: business processes modeling and measurement, software process technology and modeling, software maintenance, and methodologies for planning and managing software projects. In the past, other work topics have been: GIS (geographical information systems), educational software systems, and deductive databases. He has written 8 books and 18 chapters on the mentioned topics and published 26 papers in refereed international journals and more than one hundred papers in other journals, congresses, conferences and workshops. He has been a member of 30 program committees, reviewer in 4 scientific events, and member of 7 organizing committees. He belongs to several scientific and professional associations (ACM, IEEE-CS, ISO JTC1/SC7, EASST).

Félix García received his MS (2001) and PhD (2004) degrees in Computer Science from the University of Castilla-La Mancha (UCLM). He is currently an associate professor in the Department of Information Technologies and Systems at the UCLM. He is a member of the Alarcos Research Group and his research interests include business process management, software processes, software measurement and agile methods.

DOI: 10.1002/smr

Mario Piattini has an MSc and PhD in Computer Science from the Technical University of Madrid and is a Certified Information System Auditor and Certified Information Security Manager by ISACA (Information System Audit and Control Association). He is a professor in the Department of Computer Science at the University of Castilla-La Mancha, in Ciudad Real, Spain. Author of several books and papers on software engineering, databases and information systems, he leads the ALARCOS research group of the Department of Information Systems and Technologies at the University of Castilla-La Mancha, in Ciudad Real, Spain. His research interests are: software process improvement, database quality, software metrics, software maintenance and security in information systems.

DOI: 10.1002/smr