International Journal of Software Engineering
and Knowledge Engineering
Vol. 30, No. 5 (2020) 603–630

DOI: 10.1142/S0218194020500217

An Approach to Software Maintenance: A Case Study in Small and Medium-Sized Businesses IT Organizations

Alexandre L'Erario*,§, Hellen Christine Seródio Thomazinho† and José Augusto Fabri‡

Departamento Acadêmico de Computação (DACOM)
Federal University of Technology — Paraná
Av. Alberto Carazzai 1640 — Centro, CEP 86300-000
Cornélio Procópio, PR, Brazil
**alerario@utfpr.edu.br*
†*hellen.serodio@gmail.com*
‡*fabri@utfpr.edu.br*

Received 21 May 2019
Revised 6 October 2019
Accepted 13 November 2019

Software maintenance is the task of modifying a running product previously delivered to the client, in order to correct defects, improve performance or adapt it to the environment. This task is a crucial activity for enterprises. Without it, existing systems would become rapidly out-of-date and inefficient. The purpose of this paper is to present a software maintenance approach used in small and medium-sized business (SMB) organizations in Brazil. Currently, these organizations represent 95.5% of the software companies in the country. The approach presented here indicates how SMB IT companies have improved their software maintenance processes. Multiple case studies were performed to validate this approach. The outcomes showed that strategies associated with managing users' knowledge and development/maintenance teams are relevant to increase the maintenance process effectiveness. This approach involves three aspects: users' knowledge management, maintenance team knowledge and the management and maintenance process. This improvement includes reducing time and also minimizing the number of tickets. The response time for tickets resolution to the end user has been reduced. In addition, IT organizations have minimized the effects associated with both staff and client turnovers.

Keywords: Software maintenance; knowledge management; turnover.

1. Introduction

According to Rech and Bunse [1], the software development life cycle consists of a series of stages: planning, analysis and specification of requirements, design,

§Corresponding author.

implementation, testing, delivery and deployment as well as operation and maintenance. Upon completion of its development, the software product is delivered to the client and regardless of the application domain, size and complexity, it will continue to undergo modifications over time. Academics and software industry professionals recognize software maintenance as the most challenging and costly stage of the software life cycle [2].

Sommerville [3] emphasizes that the operational success of various organizations is directly linked to the proper functioning of systems. It may be the activity that most consumes the efforts and resources of organizations, up to 80% of the total budget according to the estimates by a study [4]. According to April and Abran [5], the cost of maintenance is very high, the speed of maintenance service is very slow and there is difficulty in managing the priority of requests for change in organizations. According to Gupta and Mishra [6], software maintenance comprises many challenges and has a high cost for organizations.

Investments in trend analysis of maintenance practice and activity planning can save much time, cost and other resources for software organization, according to Stojanov *et al.* [2]. Unfortunately, there is a lack of models to improve specific and adaptable software maintenance processes in organizations.

Besides, the problems of this activity in an organizational environment must be understood, as well as the processes to cope with the recurrence of high costs that affect organizations.

According to [7], there are more than 5000 software development companies in Brazil and 95.5% have less than 100 employees (up to 99 employees). This great majority is classified as small and medium-sized business (SMB) organizations.

The maintenance activity suggests the developer and/or IT company to retrieve old knowledge and apply this knowledge to the customer's request. Consequently, the developer sometimes needs to remember the old source code, as well as the business rules and problems in architecture maintenance, which can take a huge effort into it. Also, the customer can create a false request because of a misunderstanding during software usage.

The goal of this study is to map the approach that the SMB companies use in the software maintenance process. Based on this study, a maintenance process approach is presented to the existing software. This paper highlights the main components present in maintenance processes and the associated challenges. It also approaches the problems of these organizations including staff and customer turnovers (or churn rate). Therefore, this research tries to answer the following questions: *How do SMB organizations with user and collaborator turnovers perform the software maintenance process and how can this process be represented in a model*?

The results can be used to support organizations to create or enhance the existing maintenance processes. This work analyzes the process of ticket resolution as an important task within a software maintenance process for studied organizations. It studies different types of software maintenance tickets, namely software bug tickets and IT support tickets.

Multiple case studies were carried out to investigate the activities of the software maintenance process in Brazilian organizations. A software maintenance process model was identified, and its validation and applicability in organizations were addressed in the case studies.

The structure of this paper includes a theoretical rationale for software maintenance and related researches, presented in Sec. 2. Section 3 features the maintenance process model. In Sec. 4, the methodology is applied to the case studies, Sec. 5 highlights the analysis of outcomes and Sec. 6 comprises conclusions and future research, followed by the bibliographical references.

2. Theoretical Background

Software maintenance modifies a product previously delivered to the client to correct possible errors, enhance its performance or specific features or even adapt the product to an altered environment. Software maintenance, according to [3, 6], is a part of the software development cycle. It is not uncommon for software organizations to spend 60–70% of all their resources on maintenance [7]. According to [8], there are four types of software maintenance: corrective, preventive, adaptive and perfective.

Information Technology organizations often focus their business on software maintenance processes. This fact is a trend because it is common for organizations to concentrate resources on updating and maintaining the existing and functioning software products, rather than creating new projects [8]. Due to lack of knowledge of models and approaches that support the software maintenance process in organizations, many systems have characteristics typical of legacy systems [9, 10].

Lenarduzzi *et al.* [9] carried out a systematic literature review with the goal of analyzing the evolution of software maintenance models in the last 40 years. They investigated 1044 articles written between 1970 and 2015, including only papers published in journals and conferences in the field of Computer Science and excluding items that were not peer-reviewed or written in English. They also eliminated works that did not refer to software maintenance in the title and abstract. Based on their evaluation criteria, Lenarduzzi *et al.* [9] presented 78 articles in their systematic review. The authors concluded that software maintenance is an increasingly popular research topic, with large numbers of proposed models and approaches, according to [9].

Also, the number of proposed models is increasing, although research efforts are needed in the area to identify models that are reusable, adjustable and applicable to different contexts. Despite the increasing number of articles in this maintenance area, regarding the volume of analyzed data and complexity of models and metrics used to construct maintenance models, most are similar to those developed in the 1970s and 1980s. Due to these facts combined, the applicability of maintenance models is very limited and requires experience, according to [9], and therefore much research is still

needed to design models that are truly reusable and easy to adjust to any software and organization.

The authors [10] present challenges in the global software maintenance process in distributed projects. They investigate the factors related to software maintenance management in those settings. Their conclusion is that the challenges found in the process are related to people, product and technology.

We use the concepts of IT organization knowledge, users' knowledge and maintenance process to build our maintenance approach. In the following sub-section, these constructs are described together with some related works.

2.1. *Related works*

There are about 16,000 IT companies in Brazil involved in production, distribution and services. Furthermore, 5000 of these organizations are in the software development/production and have up to 99 employees [7]. This number has been increasing due to national demand and also due to the creation of startups.

In this scenario, many organizations take demand maintenance orders and moreover, most of them do not have quality certifications. They can be certified through the following evaluation institutions: CMMI, MPS.br (https://softex.br/mpsbr/avaliacoes/) [11] or ISO 29110 [12].

Change requests are addressed by [10, 13, 14]. In this paper, they are seen as a formal document created by customer's users that enables them to interact with the IT organization. An unclear change request can start an inappropriate maintenance development. This document is a support ticket.

Support tickets are the key to allow customers to submit their requests. According to [15], customers' problems and change request are processed as tickets. In a large company, each ticket is assigned to a support engineer for processing. Han and Sun [15] focus on large organizations and mainly how the ticket can be routed inside the company. The authors present two metrics: MSTR (Mean Steps to Resolver) and RR (Resolution Rate) and propose another one called MADR (Mean Average Distance to Resolver). This work shows a routing system supported by an expert group. As future work, they propose new routing algorithms that assist those expert groups. This approach is for large organizations and the scope of our work is for SMB organizations. SMB organizations are different from large organizations in terms of the number of employees, size of projects, customer profiles and resources.

Based on large organizations too, [16] attempts a step forward simplifying the job of supporting analysts and managers, particularly in predicting the risk of escalating support tickets. This work comprises two aspects: the first is the customers managing through handling support issues; and the second is an implementation into a machine learning model to predict the risk of ticket escalation.

The paper [17] describes practical case studies on using Trac in the development of small and medium-sized enterprise systems. This work describes practical examples of some metrics used to describe effort, size and quality, and was used for

progress management, estimation and quality assurance. This work is not aimed at customer management and aspects of staff turnover.

According to [18], software project planning addresses the activities undertaken to prepare a software development from management perspective. The activities explored are found in our case studies and mentioned by companies as very relevant for the maintenance process. Scheduling is an essential part in software development. This is the first estimation metrics and can be obtained from historical basis and other methods such as expert judgment, according to [18].

People should be allocated to the tasks, according to [18]. This book indicates the use of a matrix that shows who is responsible for what. Resource allocation (developers, in our case) is constrained by the availability. Worker allocation was explored by [19], presenting a theoretical framework for managing workers allocation in distributed projects.

According to [20], impact analysis provides information about change consequences when the changes are worth or are inevitable. There are two main aspects to impact analysis: dependency analysis and traceability.

According to [21], knowledge is essential for organizations. They propose a knowledge spiral — the conversion of knowledge from tacit to explicit. In addition, software development projects are knowledge-intensive activities [22]. The authors point out that a lot of knowledge is documented but a lot remains in the individual "mind", consequently creating an organization–knowledge gap. The paper [22] shows a guidance model for knowledge management and software engineering. It claims that in the maintenance and evolution phases there is more knowledge sharing and usage than knowledge capture and transformation. Besides, these phases require extensive collaborative work. This work does not include customer knowledge.

According to [23], customer knowledge management plays an important role in the production of high-quality software. This point of view highlights that the customers enable organizational, human and technological factors. Customer knowledge management includes the acquisition, storage, sharing and application of knowledge. The outcomes of this process are the business and operational performance, competitive advantage, innovation and product quality.

Based on socialization, externalization, combination and internalization (SECI) model for knowledge creation, explained by [24], the authors of [25] have created a set of specialized processes in each element of SECI. This work shows a list of knowledge creation processes facilitated by CRM systems. They highlight the potential of CRM for the creation of customer knowledge and point out that many companies do not explore these processes and tools.

The software operation summary presented by [26], is an approach to acquiring software operation and improving the software maintenance process. The authors [27] use the user feedback for influential software engineering tasks such as maintenance. This work assists managers in prioritizing features based on user ratings.

Business rules are often the core of software development. Understanding the business rules by the developer and the customer's ability to relate them is the starting point of a successful software project. There are lots of work done on business rules, see, for instance, [28, 29]. They focus on the relevance of the business process for the organization (customers), as well as on the importance of understanding the business process, and consequently, mapping the requirements.

It is important to highlight that the customer–IT organization relationship is managed by a formal agreement. This agreement is a set of terms and conditions that define the scope of the IT organization's work and how the customer can require changes. The service-level agreement is described by [30] as a continuous improvement process. It points out that the continuous service improvement is one of the core processes of ITIL V3.

Software maintenance is intrinsically related to many aspects. For instance, Sae-Lim *et al.* [31] show that the coding style may influence maintenance. In an exploratory research on established coding carried out from the developer's viewpoint, both participants agreed that many coding problems ended up interfering directly with software maintenance.

In addition, software maintenance is also related to software architecture. Software architecture can deteriorate over time and the cost of maintenance can increase [32]. Therefore, it is considered one of the most important aspects in software evolution [33].

Before coding, the developer needs to update his knowledge on change request and configure his development environment. This configuration can be simple or complex and can perform automatically or not. The authors [34, 35], for instance, propose automatic configuration management.

After coding and testing, the new software build (or version) can be delivered to the customer. This process can be continuous, as described by [36]. Continuous delivery can make the customer feedback faster [36].

Every change made by a developer in the source code of a product directly contributes to his or her knowledge of this software product. Such experience is essential when a new change is made in this code. In this sense, Nassif and Robillard [37] state that if no other developer in the development team is familiar with the source code of that product, the intrinsic knowledge about that product may be lost, requiring an additional effort by the organization to retrieve it. The staff turnover, according to [38], impacts the software productivity. When the developer leaves the project, this knowledge is lost. Even with adequate documentation, it is impossible to reconstruct every decision made by the developer with vast experience of the code of the software, creating significant pressure for the organization and delay in software maintenance resolutions.

Thomazinho *et al.* [39] demonstrated the application of ludic techniques in software maintenance training, with the use of robotics. There are several initiatives in the field of computing proposing the use of robots as a learning tool, mainly aiming to increase students' motivation and commitment to the development of the discipline

in question, as well as stimulating their creativity through dynamic, interactive and even ludic learning. Four experiments were carried out to validate the effectiveness of the technique described in the study, in two undergraduate courses, with the goal of measuring the team's learning capacity when applying processes and activities related to software maintenance. The results showed that the use of ludic techniques associated with robotics in the learning process proved to be relevant to motivation and satisfaction, helping consolidate concepts related to software maintenance. The authors concluded that software maintenance is linked to activities associated with the knowledge of users, the internal knowledge of IT organizations, the setup and the actual maintenance process.

There are several studies in the literature focusing on the source code, turnover, software maintenance and process improvement. This work has a combined method, emphasizing application in SMB software organizations in Brazil. Such an approach aims to cooperate with other software maintenance solutions, whether focused on users, on the maintenance process or on the team's internal knowledge.

3. Software Maintenance Process Approach

Following the review above, a set of elements relevant to software maintenance, in our scope, was listed and organized. Figure 1 shows the hierarchical organization of these elements. The elements present in the works cited in Sec. 2.1 are present here, and each element received an identification (E1, E1.1, ...) to facilitate the organization of this paper. Also in this figure, there are the references for each element.

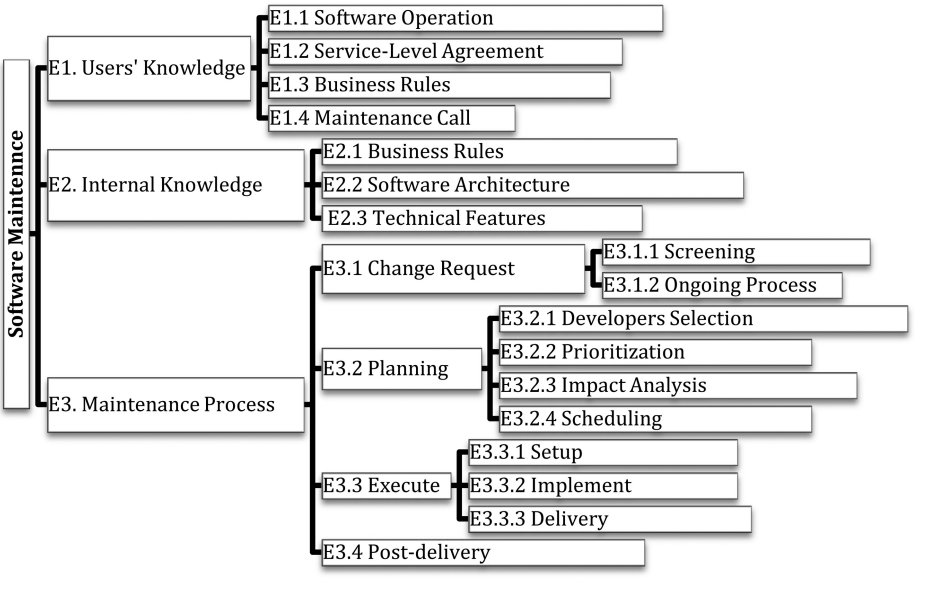

Fig. 1. The approach of software maintenance process.

The elements identified in this paper, shown in Fig. 1, cover three dimensions of software maintenance, namely: users' knowledge, internal knowledge of IT organizations and the software maintenance process. These elements are described below.

Managing users' knowledge (E1 in Fig. 1) has been identified as essential since users generate maintenance requests. The knowledge management was approached by [22, 23, 39]. The aim is to get users to correctly identify and report the problem to facilitate and streamline the software maintenance process. Users who are unaware of certain aspects of the product may request a false correction, for example. The users' knowledge process addresses the following sub-processes: knowledge of how to operate the product (E1.1, approached by [26, 27]), knowledge of contracted services (E1.2, approached by [30]) and knowledge of business rules (E1.3, approached by [28, 29]) that generates requirements. Besides, it is essential that users know how to contact the organization when they need maintenance. This element is identified in Fig. 1 as E1.4 (maintenance call, approached by [13]. The "service ticket" resource, present in many organizations, begins in this sub-process.

Internal management of the IT organization (E2 in Fig. 1) aims to manage the knowledge of the teams of staff. The knowledge of maintainers/developers is essential to solving problems, or even to intervene in improvements requested by users that are not consistent with the software product's business rules. This dimension addresses the knowledge of business rules (E2.1, approached by [28, 29]), knowledge of architecture (E2.2, approached by [19, 32, 33]) and technical knowledge (E2.3, approached by [31]).

Knowledge of business rules of the maintenance team is an essential element to get to know clients better for their performance for the market. In this sense, the focus is on business procedures executed by clients and also by the environment (restrictions or legal requirements, for example). A team that knows their clients' business rules tends to be better aligned strategically with them.

Knowledge of the product's architecture (E2.2) helps developers understand the product's operation and structure and minimize errors caused by maintenance. A developer who carries out the maintenance procedure without knowledge of the product's architecture may accidentally create another mistake. Such an event is reported in the study by [19], for example.

Managing the team's technical knowledge (E2.3) of language, programming and database also helps in assigning service tickets to specific team members, streamlining the maintenance process and generating dependency on a particular member.

Nonaka *et al.* [21] present the SECI model that shows how knowledge management can be applied in organizations. The SECI model has a spiral that displays the knowledge conversion between tacit and explicit (socialization, externalization, combination and internalization). The elements E1 and E2 of Fig. 1 indicate the problems of SMB knowledge management in software maintenance. Table 1 proposes how the SECI model [21] can be started and applied in each element namely E1 and E2. This table and Fig. 1 were applied to compose Fig. 3.

Table 1. Knowledge management in software maintenance.

	Elements	Users	Internal
E1.1	Software Operation	KS	M+
E1.2	Service-Level Agreement	M	KS
E1.3	Business Rules	KS	M+
E1.4	Maintenance Call	M	KS
E2.1	Business Rules	KS	M+
E2.2	Software Architecture	N	KS
E2.3	Technical Features	N	KS

Notes: KS: The main knowledge source. The stakeholder creates knowledge and is the main source for creating tacit knowledge.
M: The main destination of explicit knowledge.
M+: Same as M but in some cases can aggregate new tacit knowledge.
N: Nonapplicable.

The *software maintenance* process (E3 of Fig. 1) comprises the following sub-processes: (E3.1) Change request, (E3.2) planning, (E3.3) execution and (E3.4) post-delivery. The change request is the process where the user creates a ticket. Usually a ticketing tool, it can be executed from the helpdesk (user interface). After E3.1, the process E3.1.1 (screening) is executed. This sub-process may trigger the planning process (E3.2) and/or a workflow called here as ongoing (E3.1.2). This flow may occur when the actual screening detects that the request relates to a user's doubt/lack of knowledge about the product and the problem, and therefore, is solved without any changes in the software product. When product modification is required, then the planning process (E3.2, approached by [18]) is triggered by forwarding the request. This sub-process identifies the type of maintenance: corrective, adaptive, perfective and preventive.

In planning, the developers in charge of maintenance are selected (E3.2.1, approached by [9, 37, 38]), and the service ticket is sorted and assigned a priority (E3.2.2, approached by [27]). Also, it is necessary to carry out an impact analysis (E.2.3, approached by [20]) to check whether the maintenance will not cause problems in other areas of the product or agreement and finally draw up a schedule (E3.2.4). These sub-processes run in a sequential or parallel fashion.

Maintenance execution (E3.3 of Fig. 1) entails setting up the environment for development (E3.3.1), executing adequate maintenance (E3.3.2) and delivering the solution (E3.3.3) to the customer.

Setup (E3.1.1, approached by [40]) is the process in which the developer prepares his or her operating environment to execute the service ticket. Implementing is maintenance execution (E3.3.2) and involves coding and/or modifications in the database and queries as well as layout and image modifications, among others. Besides, testing and approval activities are also part of this sub-process.

Delivery (E3.3.3) involves the installation and deployment of a new version of the product for the user. Here, the sub-process may include an impact analysis related to the distribution of the new version. Post-delivery (E3.4) is the follow-up process of what has been modified in the client's product during its operation for a finite period. IT organization, based on the service-level agreement, gets the customer feedback [30] in post-delivery.

3.1. *Software maintenance process*

Figure 2 shows a software maintenance process based on the elements of Fig. 1. This process is an example of what creates a workflow among the maintenance concepts explored previously.

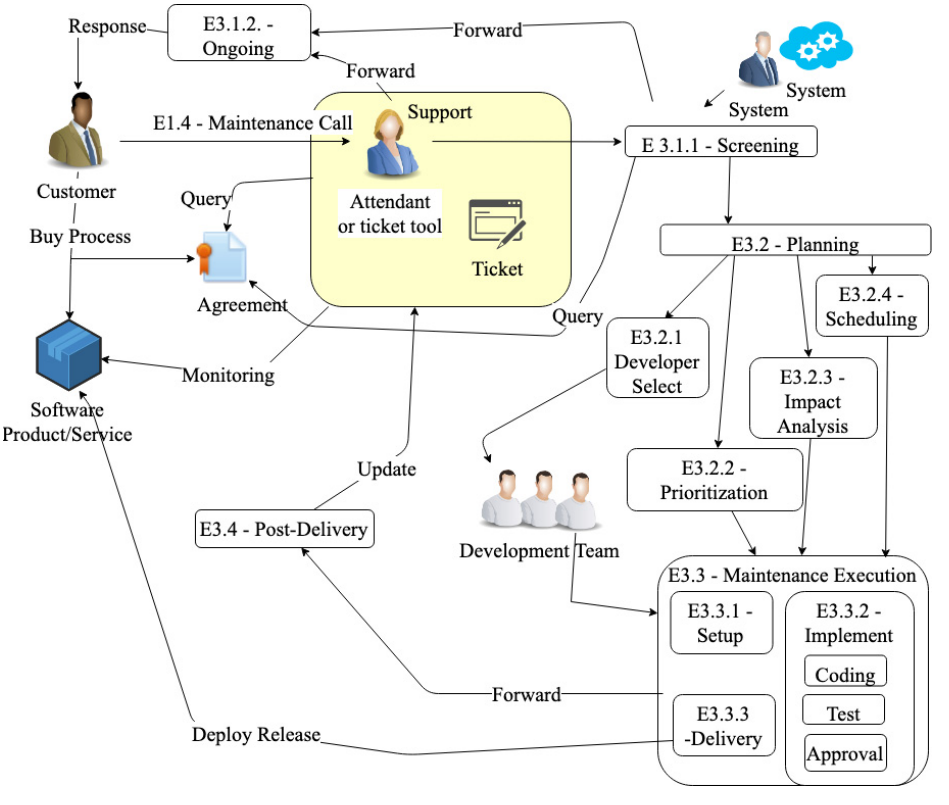

Fig. 2. Software maintenance process.

The process presented in Fig. 2 starts with customer's purchase. Although this process is out of the scope of our work, this generated two important things: a contract agreement and the use of the software.

The customer's users can interact with the IT company using the support system. This system aggregates an attendant and/or a ticket system. The support includes monitoring the use of the software by the users after delivery, according to the contract agreement. During the procurement process, the client purchases a software/maintenance package and this transaction results in an agreement.

The maintenance process may be triggered by the client or an analyst related to the system. If, on the one hand, the client can request maintenance, the systems analyst, on the other hand, can detect a potential need and request maintenance without the client's interference.

The client may request maintenance of some feature of this software product (perfective maintenance) or modification due to some product defects (corrective maintenance). For both types of request, the support generates a service ticket containing the description of the user's request/problem. The estimated term to solve it is defined in the agreement with the client. There are two possible referrals for the support. If the support itself identifies that the maintenance request is a client's doubt, it can forward the request to an ongoing workflow. The same may occur if this is detected later, in screening. The ongoing flow is a process intended to solve the client's problem without performing software modifications. Such a flow may involve more roles than those indicated in Fig. 2.

After change request and screening comes maintenance planning. The planning stage defines the developers with skills to perform the maintenance or service to identify new requirements of the software product. It is important to note that such a procedure may or may not be performed by a specific team. Regardless of the function (maintainer/developer), the service ticket triggers the maintenance process of each organization, guaranteeing the delivery of the correction or improvement to the requesting customer.

Internal and customer knowledge management are described in Fig. 3. Users' knowledge is important for the right software development once the customer business rules generate software requirements. It is important to highlight that, according to [23], the human, organizational and technological mechanisms are factors that activate the customer knowledge management and generate as outcome the enhancement of the quality of products and services. Figure 3 indicates an applied model presented by [24, 25] in our scope.

The IT company can update users' knowledge. This process occurs when the IT company updates the manuals, software documentation or interacts with users by training, email or conference, for instance. The customers can create the knowledge in some way and they can also update the documentation

A customer can have many users. For instance, one of the cases presented below is about an IT company whose main customer is a supermarket. Users turnover can occur in this case and according to [38], it is caused by lack of tacit knowledge.

The internal knowledge (Fig. 3) is the IT company staff knowledge. The development team is accountable for the results of the SMB maintenance process turnover

due to scarce resources and developers. Staff knowledge management can soften this problem through concise software documentation and guides to help them.

According to [22], most of the knowledge needed for systems maintenance has been made explicit and recorded at some point, but it has been lost. The problem of maintenance is that it is not an easy route back to the original knowledge, although documentation can help. Moreover, the same authors point out that maintenance is an intensive use of existing knowledge rather than the creation of a new one. For this reason, we believe that the manager has two problems: keeping the knowledge available for the maintenance team and selecting an ideal person for the screening process.

These previously mentioned processes strongly influence the execution and adequacy of the maintenance process. The client management process that is shown in Fig. 3 is vital since it provides support for users, making the software product more suitable to their environment and preventing them from making false maintenance

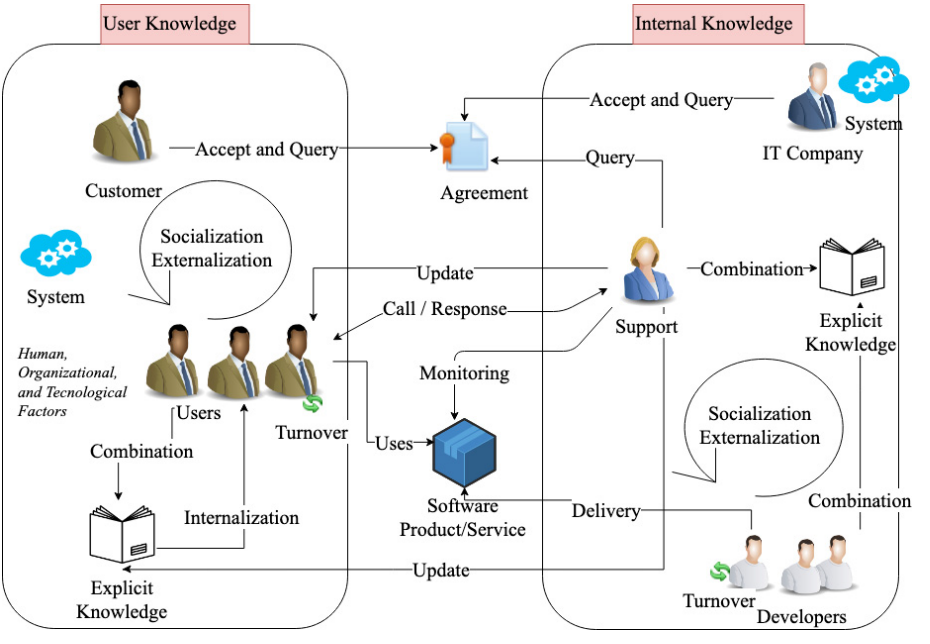

Fig. 3. Knowledge management: Customer and IT company.

requests. In this process, knowledge about the product, the contract and the business rules is disseminated and institutionalized among the stakeholders. The internal management process supports the maintenance process, thereby streamlining it and reducing errors. This process relates to business rules, software architecture and technical issues. Such concepts must be disseminated and institutionalized among the participants of the maintenance process.

4. Case Study

The method used to develop this research was making use of multiple case studies. According to Yin [41], "study is an empirical inquiry that investigates a contemporary phenomenon within its real-life context when the boundaries between phenomenon and context are not evident".

Case studies do not seek the generalization of their outcomes but rather a more in-depth understanding and interpretation of specific facts and phenomena. Although they cannot be generalized, the findings obtained should allow the dissemination of knowledge through possible generalizations or theoretical propositions that may arise from the study [41].

This work makes use of the multiple holistic designs since we applied the same research protocol to several IT companies of the Brazilian software sector and the analyzed units were the same.

To achieve the research goals of this study, we formulated the following research questions: *How do SMB organizations with user and collaborator turnovers perform the software maintenance process? How can this process be represented in a model*?

In the execution of the research protocol, after the introduction of the company and identification of stakeholders, the maintenance process was mapped and formally documented. Next, we verified whether the elements present in Figs. 1–3 are relevant to the organization. This last step not only identified the use of the model by the company but also investigated the impact caused by the absence of the elements in each case.

4.1. *Data collection*

The case study method draws on six distinct sources of information: documents, archive records, interviews, direct observation, participant observation and physical objects/objects [41]. The data sources used in this research were interviews with organizations, direct observation and artifacts available in maintenance processes. The goal of using these multiple sources of information was to verify the use of the model presented in Fig. 1 in organizations and consequently investigate not only the maintenance process but also its interface with end users (interaction between IT organizations and users regarding maintenance).

The interviews were conducted with managers and developers. Maintenance-related records (ticketing tool, notes, documents and logs) and artifacts (code, releases and builds) were also accessed.

Table 2 shows the dataset from IT companies. It is important to highlight that only the IT companies were investigated and customers were tracked by support reports. The first column in Table 2 indicates the source type, the second one the data sources and the third one indicates which elements from Fig. 1 were validated. We collected the same information from different sources, for example, the

Table 2. Research data sources and investigated elements.

Source type	Source	Investigated elements
Interview	Manager/Owner	E1, E2, E3
	Senior Developer	E2, E3
	Developer	E3.3
Direct observation	Ticket Tracking	E3
	Support Process	E1, E3.4
Artifacts	Software Analysis	E2, E3
	Source Code	E3.3.2
	Ticket Content	E1.4, E3.3.3
	Support Reports	Customer feedback, logs and notes

element E3 was investigated from many sources and validated after we made the documentation.

The manager/owner was investigated for validating all maintenance processes. It is common in small Brazilian companies for the owner to be also the manager. The senior developer is a person with leadership skills and he is usually the oldest developer with large experience. We carried out direct observation in the support process by following at least one ticket from creation to post-delivery. The content of the tickets was analyzed and in this way we can read the customer's messages and track the development. The following analysis was done in the ticket after delivery:

- *Is the request from client consistent*? We have analyzed whether the customer's modification request was coherent and made sense, considering the agreement and software usage. This analysis enabled us to identify whether the request is a mistake (lack of user knowledge of E1.1, E1.2 or E1.3) and whether the IT company executed the process E3.1.2.
- *Has the post-delivery identified the fulfillment of the request*? We have analyzed the post-delivery process. This analysis enabled us to identify the developer's execution of E3 and whether there is support that interacts with the customer after delivery.
- *Is the customer satisfied*? We have tracked the delivery quality from the customer point of view. With this analysis, we searched for problems that originated from the faulty executions of E3.2.3 and E3.2.4.
- *Was the schedule coherent*? We identified the delivery times of requests and whether the schedule was right according to the agreement.

The same aspects were analyzed from support reports and whether the IT company measures client satisfaction and provides further notes was also evaluated.

4.2. *Data analysis*

All the information collected in these case studies are confidential and therefore, we use the letters A, B, C, D and E to denote the respective organizations.

Table 3. Case descriptions.

Case	Collaborators
A	15
B	17
C	40
D	5
E	40

Table 3 identifies the organizations and the numbers of collaborators (second column of Table 3) related to development/maintenance of the software products that were investigated.

Organization A is a software manufacturer focused on developing software for the multilevel market located in the Southern region of Brazil. This organization has a maintenance process. Currently, the development team also carries out maintenance of software products and any team member is capable of performing the support of a software product. Professionals performing maintenance (development) can have direct contact with users to clarify any doubts that may arise in this process.

There is a process to manage the knowledge of all development/maintenance staff, which makes maintenance very agile. The organization is concerned about the fact that all professionals should have the same level of understanding of its products, business rules, programming languages and database, and the entire design planning provides time for professionals to improve their knowledge. This company promotes weekly workshops, and sometimes develops pair work activities when the manager deems necessary. These pair activities include support and development.

The maintenance process is started by the company's helpdesk, and the manager defines the schedule to include service tickets opened by clients, based on criticality and design, as well as on the terms to solve them. The most common types of maintenance performed are corrective and adaptive. After the screening by the manager, the impact analysis is executed by the senior developer who forwards it to a developer.

Organization B is an IT department located in the Center-West region of Brazil. It is a state government body comprising a team of 17 professionals in the area of software engineering who develop software for the state's education network. The organization has its process of software maintenance. Depending on the design, a specific team member is assigned the software maintenance. There is no significant turnover of professionals in the area, and any professional can perform software maintenance on existing products. The person in charge of maintenance has direct contact with users to clarify any doubts that may arise in this process and, in the case of preventive maintenance, meetings may be scheduled between the areas of technology and education. The software processes developed are documented and stored in a project management system that keeps track of everything that has been altered.

There is no formalized software development process in the organization, and not all team members know the programming languages used, database and others.

All software maintenance processes are managed and documented by the helpdesk, which defines the schedule to address tickets opened by clients, based on criticality and design, and also the terms to solve them. The most common types of maintenance performed are corrective and adaptive.

Organization C is a software manufacturer focused on the development of business software, located in the state of São Paulo. It has a staff of 40 professionals in the area of software engineering. The organization has its software maintenance process. The team that develops software products is also in charge of maintenance. Staff members working on maintenance have a direct relationship with the client who requested the service. Not all team members have knowledge of the programming languages used and database, and there is no stored documentation to support software maintenance. The helpdesk stores the activities recorded from the software maintenance process, which assists in opening clients' requests. The most common types of maintenance performed are corrective and adaptive.

Organization D is an information technology sector of a university in the state of São Paulo. This sector currently develops and supports all systems available in different areas of the university: academic, library, financial, legal and human resources. It comprises a team of five employees in the area of software development. The organization has its software maintenance process. The development team is also in charge of maintenance, and not all members can perform maintenance of a specific product. There is no formal and documented software development process in the organization, not all team members have knowledge of the programming languages used and database and there is no stored documentation to support software maintenance. Software maintenance requests are made by phone to the IT sector or via email, and there are no records of requests addressed, concluded or in progress. Most requests are for corrective maintenance.

Organization E is a Brazilian agribusiness software company located in the state of São Paulo. It currently develops management systems for agribusiness. The unit analyzed comprises a team of 40 employees in the area of software development. The organization has its software maintenance process. The development team is also in charge of maintenance, and not all members can perform maintenance of a specific product because product modules divide the teams and each team works only with its assigned module. Moreover, even within modules, the newest need support to perform maintenance. Within each module, all members develop and perform maintenance. Occasionally there is turnover, i.e. a member of one team can be allocated to another.

The organization has no standard for software development, and not all team members know the programming languages used and the database. Software maintenance requests reach the team via service tickets opened in the helpdesk system. Most times, the analyst in charge of the maintenance job has no contact with the client since the contact is made through customer service. In some situations, the developer contacts the user for further clarification. Most requests are for corrective and adaptive maintenance.

5. Analysis and Results

After performing the case studies, the data were tabulated. Tables 4–6 feature the tabulation of data obtained from the case studies, compared to the elements described in Figs. 1–3.

Table 4. Case study results for users' knowledge.

ID	A	B	C	D	E
E1	Exist	Exist	Inexistent	Inexistent	Exist
E1.1	Yes	No	Yes (by Hd only)	No	Yes
E1.2	Yes	Yes	No	No	Yes
E1.3	Yes	No	No	No	Yes
E1.4	Hd, Tt	Tt	Phone	Phone	Hd

Notes: Hd: Helpdesk and Tt: Ticket tool.

Table 5. Case study results for internal knowledge management.

ID	A	B	C	D	E
E2	Exist	Exist	Exist	Inexistent	Exist
E2.1	Yes	Yes	Yes	No	Yes
E2.2	No	Yes	Yes	No	Yes
E2.3	Yes	Yes	Partially	No	Yes

Table 6. Case study results for the maintenance process.

ID	A	B	C	D	E
E3	No	Exist	Partially	Inexistent	Exist
E3.1	FR	FR	Ad-hoc	Ad-hoc	FR
E3.1.1	Manager	Ticket	Expertise	Attendant	Ticket
E3.1.2	FFD	Answer	Answer	FFD	FFD
E3.2	Existent	Existent	Partial	Partial	Existent
E3.2.1	AD	AD	ED	AD	AD
E3.2.2	Yes	Yes	No	No	Yes
E3.2.3	Yes	Yes	No	No	No
E3.2.4	1 week	4 days	5 days	5 days	2 days for corrective
E3.3	Developer	Maintenance team	Developer	Developer	Developer
E3.3.1	10 min	60 min	Un	Un	Un
E3.3.2	Coding/Approval	Coding/Test	Coding	Coding	Coding/Test/Approval
E3.3.3	Developer	Support/Approval by user	Developer	Developer	Developer

Notes: FR: Forward request; FFD: Forward to feature developer; AD: Available developer; ED: Expertise developer; and Un: Uninformed.

5.1. *Users' knowledge*

The users' knowledge (E1) was explored in the five cases and shown in Table 4. Considering the row E1, which aimed to gather information on users' knowledge

management, we identified whether the organizational support structured users' knowledge management.

The organizations A, C and E have the process concerning software operations knowledge (E1.1). There is an inherent concern in these cases regarding providing the clients with knowledge (through manuals, texts, training) on product operation. Company A has created a user-friendly webpage to explain what business rules the software follows. The company believes that it has been a great improvement in its process. Due to this webpage, the numbers of false change requests from their customers and incorrect sales have been reduced.

The main customer of case C is a supermarket, and the process E1.1 has been executed by helpdesk phone only. According to the company, there is a set of complex business rules needing support, whereas the other business rules are simple. To aid the set of complex business rules, a helpdesk with a phone is enough. Case E creates a manual for each functional requirement and notifies the customer.

Cases B and D do not provide such input to clients, and in these cases, they have some difficulty in operating the software product. This lack resulted in a large number of false requests, according to these cases. Providing knowledge of how to operate software minimizes the number of false requests made by users to the call center. Also, it fosters a better understanding of the technical aspects of the product's operation, which makes it more user-friendly and simpler.

Element E1.2 informs clients of the limitations regarding requests for modification/update and help with contracted products/services. This element indicates the contracted scope. Knowledge of contracted services/products (E1.2) is present in the cases A, B and E, and in these cases, there is a better understanding of users regarding the service package agreement. Service package in this work is the agreement between IT company and client regarding the use and maintenance/update of software. Regarding organization B, as it operates in the public sector there is no service agreement; neither is there one in organization D, since it operates inside the actual customer (onshore). There is a standard agreement available at organization C, and the customer understands the terms of the contract. However, this case reported that users created false change requests because of misinterpretations.

The element E1.3 (business rules) exists only in organizations A and E. Software users are aware of the business rules of the software product acquired and consequently of their requirements. It was noted in cases A and E that when users are aware of the business rules, the requirements passed on to the organization from the client are more efficient. The client is benefited from adequate product deployment, and the IT organization minimizes faulty implementations due to customer requirements.

All organizations have some methods to start service tickets (E1.4) of requests made by users. Organizations A, B, C and E have a ticketing tool whereas organization D has no mechanism to control service tickets. In the case of organization C, an attendant creates the ticket. In the case of organizations A, B, C and E,

maintenance documentation and history are stored in (or associated with) the actual ticketing tool. For case D, an attendant manages the request.

The E1 element and its sub-processes were found to be highly significant for:

- promoting the effectiveness and efficiency of the contracted service/product;
- making the maintenance execution straightforward and faster by disseminating knowledge about business rules among all users;
- avoiding the opening of false service tickets by the users;
- solving quickly as to whether the service tickets are related to maintenance or not;
- creating records of interactions between IT organizations and their clients;
- outlining and/or enabling readjustment of the scope of the agreement between clients and IT organizations.

5.2. *Internal knowledge*

Table 5 shows the internal knowledge management. This is the element E2 of Fig. 1.

The E2 element intended to identify aspects related to managing knowledge of IT maintenance staff. The E2.1 sub-process, related to knowledge of business rules, was only absent in case D. All other cases showed an intrinsic concern with disseminating the operation of the client's business rules among IT staff, with all developers involved in the product in question knowing such rules. These were often communicated through training and workshops. Knowledge of business rules by developers broadens the options of staff involved in potential maintenance processes. In this sense, managing knowledge of clients' business rules within the IT organization minimizes dependency on specific developers.

Regarding the aspects of architecture (E2.2), organizations A, B, C and E manage knowledge related to software architecture. In organization D, only one member of the team possesses architecture-related skills. Knowledge of architecture among developers prevents them from degrading other aspects of the software product when executing maintenance procedures.

Analyzing E2.3, which aims to gather information on aspects related to managing the technical knowledge of maintenance staff, this element is only absent in organization D, since not all members have at least the technical knowledge of the languages in which the software products were developed to perform proper maintenance. The case C focuses its effort only on the few more complex rules that are executed by the software, and as far as the other rules are concerned, the company judges that it is easy for the developer to understand them in a change request.

In all cases element E2 and its sub-processes were identified as relevant for:

- facilitating developers' understanding of requests made by the clients;
- minimizing the dependency of the IT organization on a specific developer;
- enabling several developers to perform maintenance procedures;

- minimizing negative impacts (degrading other features, for example) during maintenance;
- providing input for improved maintenance planning (element E3.2);
- minimizing setup time (E3.3.1) by the developers;
- improving communication between developers (executing maintenance) and clients.

5.3. *Maintenance process*

Table 6 shows the maintenance process for each case. This is the element E3 of Fig. 1. All organizations reported that after or during customer contact (which may be via online ticketing tool, helpdesk or telephone, for example), a first screening process is performed. Fig. 1 indicates this process.

The support analyst or ticketing manager is in charge of the helpdesk that executes the screening process in organizations A, B, C and E. This person checks whether it is a case of doubt or correction/maintenance. In the case of organization D, there is no person responsible for this role, since most requests are made by telephone to the sector, and the call is put through to the developer in charge of the feature in question.

There are two approaches to the screening process. The first one occurs when the user's request for maintenance is justified, i.e. the software needs modifications. In this case, the maintenance process begins with the request being forwarded to the next step/person in charge. However, the client's request may be identified as a doubt, leading to the so-called ongoing flow (E3.1.2). The user will receive an answer to his or her problem without execution of the maintenance procedure (modify/update the software). Sub-process E3.1.1 (screening) was detected in all cases.

The planning sub-process (E3.2 of Fig. 2) consists of the following sub-processes: choice of developers (E3.2.1), prioritization (E3.2.2), impact analysis (E3.2.3) and scheduling (E3.2.4).

In organizations A, C and E, all developers are capable of performing maintenance, and therefore any staff member may be chosen for this task. In organizations B and D, maintenance is also performed by the development team, depending on the software design, and in these cases, there is a specific professional solely to perform maintenance.

Employee turnover was identified only in organizations A and C. According to the organization C, this affects software maintenance significantly, because in some cases knowledge of the software business rules is focused on users, which can cause problems for the IT organization. Case A reported some developer turnover, but after the organization has been improving its E2 process, turnover happened but it did not have any problem. In the other cases, this problem was not identified.

Regarding the prioritization of maintenance tasks (E3.2.2), organizations A, B and E execute this sub-process while in the other organizations it is nonexistent.

In cases C and D, this sub-process is absent because the client deliberates prioritization. In this sense, internal management of employee allocation occurs on demand.

Impact analysis (E3.2.3) of maintenance was observed in organizations A and B. This analysis is essential for developers to perform maintenance without affecting other modules/features of the software product (dependence analysis). In other cases, this sub-process was reported as nonimportant and/or was executed during the actual maintenance performed by the developer. In cases C and D, there were reports of developers who accidentally affected another module of the product. Case E allocates an expert (the developer who knows the business rule and the technology) to perform maintenance, thus minimizing the problem.

Organization A reported that corrective maintenance takes one week at most (E3.2.4) while the term for adaptive maintenance depends on the agreement with the client. Organization B reported that the term for corrective maintenance is up to four days depending on the demand and adaptive maintenance has no estimated term for the conclusion. In organization C corrective maintenance is carried out in up to five days, depending on the problem. In organization D the term depends on the developer's activities; urgent cases may be done in up to five days, but there is no actual estimate. In organization E, in turn, corrective maintenance is performed in up to two days.

In organization A, a developer in charge of the development team performs maintenance (E3.3 in Fig. 1). In organization B, there is a specific analyst who only does maintenance. In organizations C, D and E, it is done by the developer with a high level of knowledge of the product.

Setup management (setup element, indicated as item E3.3.1 in Fig. 2) investigates the time/complexity involved in setting up the work environment to start maintenance. In the organization A, setup time is around 10 min and in organization B, it is 1 h, while organizations C, D and E were unable to determine the time needed. Only case A had specific setup management documentation that enabled developers to set up their environments. In the other cases, all developers kept the product previously set up in their environments.

Following environment setup (E3.3.1 in Fig. 1), the organizations would execute the actual maintenance. This process is indicated as *implement* (E3.3.2) in Fig. 1. The implementation sub-process may involve several activities: coding, layout adjustments, testing and homologation. All organizations claim that users may be contacted during coding. In organization B, communication is done by the maintainer and in the other organizations by the actual developer. It was observed in the studies that contact with clients during this stage facilitates maintenance conclusion since many times the service ticket is not clear and raises doubts regarding the system's operation. Such interaction often occurs in the organizations, whether via helpdesk, telephone or even email.

In organization B, the actual maintainer performs testing. The developer performs tests in the companies C, D and E. Homologation of the new version after

maintenance was identified in organizations B and E. In the other organizations the new version goes directly into production.

Regarding the delivery sub-process (E3.3.3), in organization A the client is notified that there is a version already available for homologation and delivery is made by the developer. The previous version remains in the production environment until the version in the client's homologation environment is approved. Approval/acceptance by the client happens within five days.

In organization B, the person who opened the service ticket receives a notification via helpdesk that there is a version available in the homologation environment. The analyst who performed the maintenance also updates both environments (homologation and production) and performs the 48-h follow-up with the client. In organization E, the client is also notified via helpdesk about the availability of a new version. The developer in charge of maintenance updates the client's environment and follow-up with the client is done over 24 h. In organizations E, C and D, delivery and follow-up are the responsibilities of the developer.

The organizations A, B and E execute post-delivery follow-up (item E3.4 of Fig. 1). The other organizations do not have a follow-up process.

All cases make direct contact with the requesting user of the service ticket during maintenance. Such contact helps in solving doubts and identifying the objectives of new requirements or even the operating feature that caused a specific software error.

The results obtained in the case studies showed that the investigated organizations perform processes that are similar to those shown in Fig. 2.

Also, two other processes strongly influence the execution and adequacy of the maintenance process. The client management process, shown in Fig. 3, is vital for it provides support for users, making the software product more suitable to their environment and preventing them from making false maintenance requests. In this process, knowledge about the product, the contract and the business rules is disseminated and institutionalized among the stakeholders. The internal management process, shown in Fig. 3, supports the maintenance process, thereby streamlining it and reducing errors. This process relates to business rules, software architecture and technical issues. Such concepts must be disseminated and institutionalized among the participants of the maintenance process.

5.4. *Analysis results*

After we analyzed the case studies, we figure out that many times maintenance is focused on a single developer (more experienced) due to the lack of knowledge of the other team members, creating a maintenance relationship with only one professional and hindering the team from streamlining the process. It can be concluded that all the organizations investigated perform a pre-established software maintenance process, even when there is no formalization and/or documentation.

None of the organizations was aware of the standards related to maintenance regarding specific maintenance models, and none of the organizations showed

incentive to implement the models MPS.br [11], ISO 29110 [12], CMMI or another, although some adherence to them was observed.

From the organizations, there is no differentiation between the types of maintenance. Following the screening process, the service ticket is executed within the development process with no distinction. Differentiation between corrective maintenance and the other types may be more evident in agreements between clients and the IT company.

The cases with poorly formalized processes that did not implement internal and clients' knowledge management (cases C and D), according to Fig. 1, reported having high maintenance costs. These cases are studying the implementation of such processes precisely to minimize costs and improve customer service.

Based on these cases, we can conclude that the E1 (users' knowledge) process relevance depends on both the amount and complexity of business rules. This relevance can be high, if the software has many complex software rules, with unattached number of users. Cases A, B and E are included in this approach. However, if the company considers the business rules simple (cases C and D), then the users' knowledge is less important. Figure 4 features the cases. It is important to emphasize

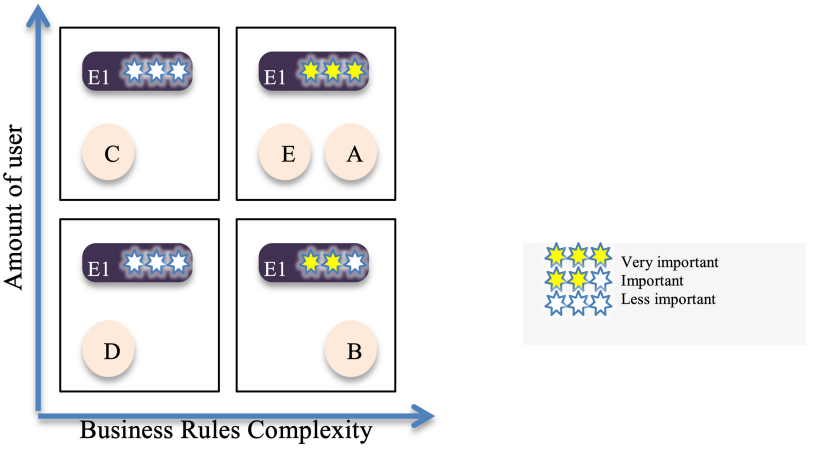

Fig. 4. Importance of E1 to the companies.

that case A has been improving its process after we developed this approach. This process has been reducing the customer and user mistakes caused by the lack of software operation and by mainly the misunderstanding of complex business rules.

Case D reports an onshore organization and for this reason, the need of E1 process is low.

E2 is similar to E1. However, we have detected that the architecture influences the internal knowledge (see Fig. 5). If the architecture increases with nonfunctional requirements according to customer agreement or unstable baseline, the companies

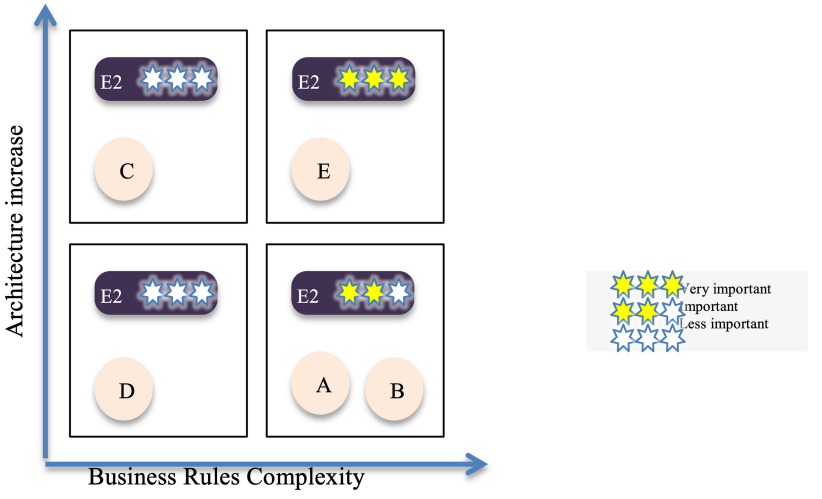

Fig. 5. Importance of E2 to the companies.

find process E2 more relevant (the cases B, C and E are examples). However, the knowledge about technical features was considered not important for cases C and D. Case C can aggregate more nonfunctional requirements in its software, although there is a large set of simple business rules. The company claims that it can be easily maintained by a developer. Case D does not aggregate nonrequirements in its software and considers that any developer can maintain it. Before the developers start their job, case A provides a large user-friendly material for the collaborators so that they can understand the architecture and the complex business rules.

5.5. *Threats to validity*

According to [42], the aspects of validity can be summarized as construct validity, internal validity, external validity and reliability.

The internal validity threat was minimized by cross-checking data from multiple sources (documents and interviewers). Besides, we explain each aspect of our approach (Figs. 1 and 2) for the interviewers.

The external validity aspect is concerned with to what extent it is possible to generalize the findings, according to [42]. We believe that our results can be applied in other organizations since there is a lot of SMB IT organizations in Brazil and perhaps many of them with the same problems.

Validity threats may affect the reliability of the results. We detected the following aspects:

(1) Selection of interviewers: The interviewees were colleagues of the authors who were working in companies. Besides, at least two interviewees were selected for

each company. The process specification and the elements in Figs. 1–3 were documented and the interviewer validates afterwards.

(2) Direct observation: At least one charge request was tracked. We followed the execution of the maintenance process from after ticket creation until delivery. We did not contact the customers.

(3) Artifacts: In order to minimize the misunderstanding of artifacts, we analyzed some of them together with collaborators.

(4) Generalizability: This is a typical threat to validity in case studies. We conducted case studies in five SMB IT companies. This sample is small if compared to the number of companies in Brazil (more than 5000 SMB organizations). However, very detailed and deep information was gathered. Furthermore, according to [43], we can generalize to other cases, though similarity is a weak basis for generalization. We believe that there are many companies in Brazil with the same aspects. For instance, in the Technology Incubator of UTFPR (Federal University of Technology of Paraná), we have found other two companies with the same problems.

6. Conclusions

This paper featured the case study of some organizations whose software maintenance problems are aggravated by the lack of a model for activities and tasks within a business context. It was shown that such problems directly interfere with the software maintenance activity, resulting in lack of planning, high costs, task overloading and failure to communicate with the users of those systems.

Therefore, based on the analysis presented in this case study, it was possible to answer the following questions: *How do SMB organizations with user and collaborator turnovers perform the software maintenance process and how can this process be represented in a model*?

In all organizations, the development team itself performs software maintenance. None of the organizations has a specific team dedicated to maintenance activities, but they have not ruled out the possibility of having such a team and that such a possibility exists according to the agreement.

The screening process depends on the software process and to what extent the knowledge of users and development teams is institutionalized. The better the maintenance process is structured and the knowledge described in Fig. 3 is institutionalized, the less complex and shorter the screening process will be. It was noted that the absence of formalized processes and knowledge management of users and staff caused a more experienced developer to execute the screening procedure, i.e. many times the developer had divided his or her attention between a current design and a maintenance request. In addition, it was reported that such a procedure became much more costly because for each request the more experienced developer had to integrate the users' knowledge (check if it was actually a maintenance request and verify the functionality of the business rule) with that of his or her team

(indicate where maintenance would be performed and explain the business rule and software architecture, for example) for the maintenance to be performed successfully.

This study made it possible to perceive the flaws in the software maintenance process of the organizations investigated, since it also enabled a view of what needs to be improved to perform software maintenance successfully and deliver the product to the client within the estimated term, and with better quality.

We conclude that organizations can meet maintenance demands, whether formal or informal, but that creating a process shown in Figs. 1 and 2 and deploying the knowledge management processes shown in Fig. 3 tend to serve clients better and reduce costs related to software maintenance.

Based on the research, it observed that professionals prefer to have proactive improvement strategies. It is concluded that the data mining and predictive analysis process provides a combination of valuable information applicable to various software maintenance areas, ranging from defect detection and resolution to IT support ticket resolution processes.

The approach herein presented addresses all types of maintenance. However, in the cases investigated there were no reports of preventive and perfective maintenance. In this sense, as future research, a new case study will be applied to an organization that performs these two types of maintenance with the purpose of validating the model as a whole.

It is suggested as future research to deploy this model in organizations and carry out a survey of staff and management to validate the efficiency of the proposal.

Acknowledgments

We would like to thank the Brazilian companies for their participation in this study.

References

1. J. Rech and C. Bunse, *Emerging Technologies for the Evolution and Maintenance of Software Models*, 1st edn. (IGI Global, Hershey, 2011).
2. Z. Stojanov, J. Stojanov, D. Dobrilovic and N. Petrov, Trends in software maintenance tasks distribution among programmers: A study in a micro software company, in *Proc. 2017 IEEE 15th Int. Symp. Intelligent Systems and Informatics*, 2017, pp. 23–28.
3. I. Sommerville, *Software Engineering*, 9th edn. (Addison-Wesley, 2010).
4. J. Li *et al.*, Cost drivers of software corrective maintenance: An empirical study in two companies, in *Proc. 2010 IEEE Int. Conf. Software Maintenance*, 2010, pp. 1–8.
5. A. April and A. Abran, *Software Maintenance Management: Evaluation and Continuous Improvement (Practitioners)* (Wiley-IEEE Computer Society, Hoboken, 2008).
6. G. Gupta and R. P. Mishra, A SWOT analysis of reliability centered maintenance framework, *J. Qual. Maint. Eng.* **22**(2) (2016) 130–145.
7. ABES, Associação Brasileira das Empresas de Software, Brazilian Association of Software Companies, 2018, http://central.abessoftware.com.br/Content/UploadedFiles/Arquivos/Dados%202011/af_abes_publicacao-mercado_2018_small.pdf.

8. ISO, Standard 14764-2006 — ISO/IEC/IEEE standard for software engineering: Software life cycle processes — Maintenance, Revision of IEEE Standard 1219-1998, 2006, pp. 1–46.
9. V. Lenarduzzi, A. Sillitti and D. Taibi, Analyzing forty years of software maintenance models, in *Proc. 2017 IEEE/ACM 39th Int. Conf. Software Engineering Companion*, 2017, pp. 146–148.
10. B. Ulziit, Z. A. Warraich, C. Gencel and K. Petersen, A conceptual framework of challenges and solutions for managing global software maintenance, *J. Softw. Evol. Process* **27**(10) (2015) 763–792.
11. M. A. Montoni, A. R. Rocha and K. C. Weber, MPS.BR: A successful program for software process improvement in Brazil, *Softw. Process Improv. Pract.* **14**(5) (2009) 289–300.
12. R. V. O'Connor and C. Y. Laporte, Software project management in very small entities with ISO/IEC 29110, in *EuroSPI 2012: Systems, Software and Service Process Improvement*, Communications in Computer and Information Science, Vol. 301 (Springer, 2012), pp. 330–341.
13. I. Bassey, Enhancing software maintenance via early prediction of fault-prone object-oriented classes, *Int. J. Softw. Eng. Knowl. Eng.* **27**(4) (2017) 515–537.
14. A. A. Alsanad, A. Chikh and A. Mirza, Multilevel ontology framework for improving requirements change management in global software development, *IEEE Access* **7** (2019) 71804–71812.
15. J. Han and A. Sun, Mean average distance to resolver: An evaluation metric for ticket routing in expert network, in *Proc. 2017 IEEE Int. Conf. Software Maintenance and Evolution*, 2017, pp. 594–602.
16. L. Montgomery and D. Damian, What do support analysts know about their customers? On the study and prediction of support ticket escalations in large software organizations, in *Proc. 2017 IEEE 25th Int. Requirements Engineering Conf.*, 2017, pp. 362–371.
17. N. Ohsugi *et al.*, Using Trac for empirical data collection and analysis in developing small and medium-sized enterprise systems, in *Proc. 2015 ACM/IEEE Int. Symp. Empirical Software Engineering and Measurement*, 2015, pp. 1–9.
18. IEEE Computer Society, *Guide to the Software Engineering Body of Knowledge (SWEBOK (R)): Version 3.0*, eds. P. Bourque and R. E. Fairley (IEEE Computer Society Press, Washington, DC, 2014).
19. R. Bonet and F. Salvador, When the boss is away: Manager–worker separation and worker performance in a multisite software maintenance organization, *Organ. Sci.* **28**(2) (2017) 244–261.
20. M. Shahid and S. Ibrahim, Change impact analysis with a software traceability approach to support software maintenance, in *Proc. 2016 13th Int. Bhurban Conf. Applied Sciences and Technology*, 2016, pp. 391–396.
21. L. Nonaka, H. Takeuchi and K. Umemoto, A theory of organizational knowledge creation, *Int. J. Technol. Manage.* **11**(7–8) (1996) 833–845.
22. J. B. de Vasconcelos, C. Kimble, P. Carreteiro and Á. Rocha, The application of knowledge management to software evolution, *Int. J. Inf. Manage.* **37**(1) (2017) 1499–1506.
23. A. Khosravi, A. R. C. Hussin and M. Nilashi, Toward software quality enhancement by Customer Knowledge Management in software companies, *Telemat. Inform.* **35**(1) (2018) 18–37.
24. I. Nonaka, A firm as a knowledge-creating entity: A new perspective on the theory of the firm, *Ind. Corp. Change* **9**(1) (2000) 1–20.

25. F. Khodakarami and Y. E. Chan, Exploring the role of customer relationship management (CRM) systems in customer knowledge creation, *Inf. Manage.* **51**(1) (2014) 27–42.
26. H. van der Schuur, S. Jansen and S. Brinkkemper, Sending out a software operation summary: Leveraging software operation knowledge for prioritization of maintenance tasks, in *Proc. 2011 Joint Conf. 21st Int. Workshop on Software Measurement and the 6th Int. Conf. Software Process and Product Measurement*, 2011, pp. 160–169.
27. J. Zhang, Y. Wang and T. Xie, Software feature refinement prioritization based on online user review mining, *Inf. Softw. Technol.* **108** (2019) 30–34.
28. G. J. Nalepa and K. Kluza, UML representation for rule-based application models with XTT2-based business rules, *Int. J. Softw. Eng. Knowl. Eng.* **22**(4) (2012) 485–524.
29. J. Silvander, M. Wilson, K. Wnuk and M. Svahnberg, Supporting continuous changes to business intents, *Int. J. Softw. Eng. Knowl. Eng.* **27**(8) (2017) 1167–1198.
30. V. Marković and R. Maksimović, A contribution to continual software service improvement based on the six-step service improvement method, *Int. J. Softw. Eng. Knowl. Eng.* **22**(4) (2012) 549–569.
31. N. Sae-Lim, S. Hayashi and M. Saeki, How do developers select and prioritize code smells? A preliminary study, in *Proc. 2017 IEEE Int. Conf. Software Maintenance and Evolution*, 2017, pp. 484–488.
32. F. Schmidt, S. MacDonell and A. M. Connor, Multi-objective reconstruction of software architecture, *Int. J. Softw. Eng. Knowl. Eng.* **28**(6) (2018) 869–892.
33. S. Kang and D. Garlan, Architecture-based planning of software evolution, *Int. J. Softw. Eng. Knowl. Eng.* **24**(2) (2014) 211–241.
34. E. Luchian, C. Filip, A. B. Rus, I.-A. Ivanciu and V. Dobrota, Automation of the infrastructure and services for an openstack deployment using chef tool, in *Proc. 2016 15th RoEduNet Conf.: Networking in Education and Research*, 2016, pp. 1–5.
35. N. P. N. Xuan, S. Lim and S. Jung, Centralized management solution for vagrant in development environment, in *Proc. 11th Int. Conf. Ubiquitous Information Management and Communication*, 2017, pp. 1–6.
36. M. Shahin, M. Ali Babar and L. Zhu, Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices, *IEEE Access* **5** (2017) 3909–3943.
37. M. Nassif and M. P. Robillard, Revisiting turnover-induced knowledge loss in software projects, in *Proc. 2017 IEEE Int. Conf. Software Maintenance and Evolution*, 2017, pp. 261–272.
38. E. Oliveira, T. Conte, M. Cristo and N. Valentim, Influence factors in software productivity — A tertiary literature review, *Int. J. Softw. Eng. Knowl. Eng.* **28**(11–12) (2018) 1795–1810.
39. H. C. S. Thomazinho, A. L'Erario and J. A. Fabri, Teaching software maintenance with ludic techniques supported by robotics, in *Proc. 2017 IEEE Frontiers in Education Conf.*, 2017, pp. 1–8.
40. E. D. Farahani and J. Habibi, Configuration management model in evolutionary software product line, *Int. J. Softw. Eng. Knowl. Eng.* **26**(3) (2016) 433–455.
41. R. K. Yin, *Case Study Research and Applications: Design and Methods* (Sage Publications, 2017).
42. C. Wohlin *et al.*, *Experimentation in Software Engineering* (Springer, Berlin, 2012).
43. R. Wieringa and M. Daneva, Six strategies for generalizing software engineering theories, *Sci. Comput. Program.* **101** (2015) 136–152.