Software maintenance improvement in small software companies: Reflections on experiences

Zeljko Stojanov

University of Novi Sad, Technical faculty "Mihajlo Pupin" Zrenjanin, Serbia

E-mail: zeljko.stojanov@uns.ac.rs

Abstract. Software maintenance has been recognized as the most demanding and costly phase in the software life cycle. Software maintenance tasks, although require a more complex set of skills and knowledge, are far less interesting to software engineers than software development tasks. In addition, insight into the scholarly literature revealed that the knowledge basis on software maintenance is significantly less than the knowledge base on software development. Due to the obvious constraints of small software companies, they do not have time, people, and other resources for assessing and improving their software maintenance practice. This paper presents the author's reflections on experience in assessing and improving software maintenance practice in an indigenous micro software company.

1. Introduction

Contemporary market and business require software organizations capable to quickly develop and effectively maintain complex software systems. Although software development has been much more attractive to software engineers than software maintenance, software maintenance has been recognized as the most costly, demanding and difficult phase in the software life cycle [1, 2, 3, 4]. By analyzing maintenance models used more than forty years in software industry, Lenarduzzi et al. [5] indicated that: (1) majority of models were built from scratch without extending or using existing ones, (2) developed models were proposed for specific problems, which makes them difficult to compare to other models, and (3) models were validated only by people that propose them, which puts some limitations on them regarding their effectiveness.

Software systems suitable for maintenance should satisfy several customers criteria such as the quality, reliability, user-friendliness, and technical criteria of the domain of use, but also they should align with business processes and strategic orientation of a software organization that provide maintenance services. With recognized maintenance costs and the complexity of maintenance activities, development of maintainable software systems is one of the most demanding and important requirements for the software industry [6]. In addition, the major problem with software maintenance in software industry is that many software organizations do not have defined maintenance processes, which includes the lack of process models and process management [7]. These observations suggest that software organizations should strive to continuously improve software maintenance processes. Maintenance practice should be observed

and measured in order to propose the most suitable methods and tools for solving maintenance problems, which requires empirical validation in variety of industrial settings. Juergens [4] argued that successful software maintenance practice requires work from both practitioners and researchers perspective, with full support for feedback from the industrial practice.

The rest of the paper is structured as follows. The second section outlines the basic concepts and trends in software maintenance, while the third section presents a short insight into software engineering practice in small software organization with the focus on their processes improvement. The fourth section presents the authors reflections on experience within a project related to software maintenance practice implemented in a micro software company in Serbia. The conclusions are presented in the lasts section.

2. Software maintenance

Despite the proved complexity and high costs of maintenance activities, maintenance has received far less attention than the other phases in software life cycle. However, nowadays software organizations try to get as much benefits from developed software as possible by keeping them operating for a long period of time [2]. According to the international standard *ISO/IEC/IEEE 14764-2006: Software Engineering - Software Life Cycle Processes - Maintenance* [8], software maintenance is defined as the totality of activities required to provide cost-effective support to a software system, mainly in post-delivery phase of software life cycle. Software maintenance process is a set of activities that enable modifications of the existing software product while preserving its integrity until it is finally retired. In most cases, software maintenance does not involve making major modifications to the architecture and the design of a software system, but rather smaller modifications that keep deployed software system useful and aligned with the users' need. [9]. The objectives of software maintenance are [2]: (1) to sustain day-to-day functioning of software systems, (2) to control modifications of software systems, (3) to improve existing functions in software systems, (4) to identify and fix security issues, and (5) to prevent degradation of software performances.

According to the standard *ISO/IEC/IEEE 14764-2006*, the main activities that comprise software maintenance process are [8]: (1) process implementation - establishing the plans and procedures for process activities, (2) problem and modification analysis, (3) implementation of a modification - developing and testing the modification, (4) maintenance review/acceptance - ensuring the correctness of the implemented modification, (5) migration to new environments, and (6) retirement - decision that software is not useful based on the analysis. An overview of a maintenance process with a cycle that my include several iterations of problem and modification analysis, implementation of a modification, and maintenance review/acceptance activities is presented in figure 1.

Based on the work of Haworth et al. [10], Grubb [11] proposed a *Maintenance framework* that contains all factors that affect maintenance practice:

- *User requirements* relates to requests that include modification of software such as adding functionality, correcting errors or improving maintainability, as well as requests for non-programming-related support.
- *Organizational environment* relates to internal policies and organizational structure, external regulatives, as well as competition in the market place.
- *Operational environment* relates to technical infrastructure in which software is integrated, including other software systems, hardware components, and other domain-specific components (e.g. monitoring equipment in industrial systems).
- *Maintenance process* relates to understanding user requirements through creative work with undocumented assumptions, dealing with differences in the approaches used for developing and maintaining programs, and detecting and correcting errors.

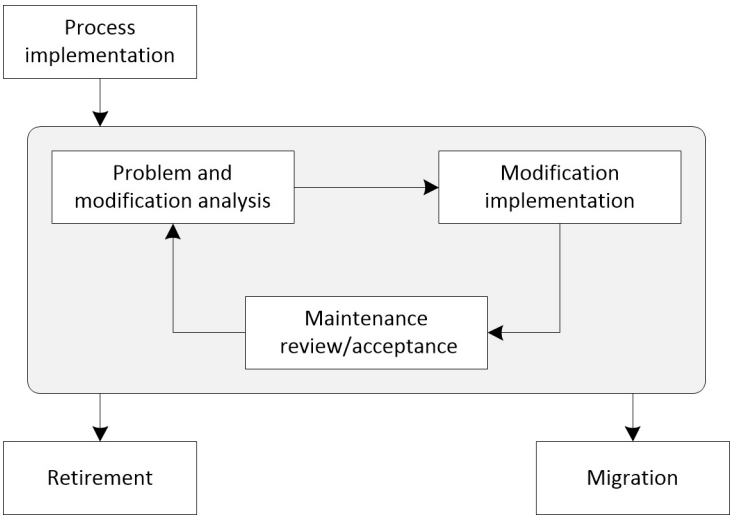

Figure 1. Maintenance process overview (adapted from [8]).

- *Software product* relates to maturity and difficulty of the application domain, quality (or the lack) of documentation, complexity and flexibility of program structure, and changes in software quality.
- *Maintenance personnel* relates to the high staff turnover (the question is whether the same people do development and maintenance tasks), and domain expertise when moving from one to another software.

Identified factors within the maintenance framework interrelate, which causes maintenance problems and as a consequence evolution of a software product. The major interactions within software maintenance occur between software product and environment, software product and software user, and software product and maintenance personnel. Software product is used within technical and organizational environments and should evolve to remain useful if something change within the environment. The needs of software users change as change their business, causing software systems to change to stay aligned with their changed needs. And finally, maintenance personnel is responsible for software modifications, so that their characteristics significantly affect the technical details of modifications, the quality of changed software, and the way of maintenance process implementation.

The analysis of the factors that affect the work efficiency of software maintenance based on company data sets from 120 organizations revealed that the high difficulty level of the libraries, compilers, test tools, maintenance tools, and reverse engineering tools has negative impact on maintenance work efficiency, while higher costs of staff do not have negative effects on maintenance work efficiency and software quality [12]. These findings indicate that the technical factors of the work have a more significant impact on maintenance efficiency than issues related to maintenance staff.

In software maintenance practice occur different types of maintenance that have been arranged within maintenance typologies. The first, intention based typology was proposed by Swanson [13], which includes *corrective maintenance* (reactive actions to correct discovered problems), *adaptive maintenance* (modification aimed at keeping software usable in changed environment), and *perfective maintenance* (enhancements of software products). This typology has been extended with *preventive maintenance* (identification and correction of latent problems in software systems) within the standard *ISO/IEC/IEEE 14764-2006: Software Engineering* -

Software Life Cycle Processes - Maintenance [8]. Chapin [14] extended the typology proposed by Swanson into evidence based typology in which software maintenance types are determined based on a set of question related to the effect of maintenance activities on software products. This extended maintenance typology contains 12 types of software maintenance: training, consultive, evaluative, reformative, updative, groomative, preventive, performance, adaptive, reductive, corrective, and enhancive. Software maintenance also includes support activities (*supportive maintenance*) that do not require modification of software, such as non-technical help desk, support for quick solution of technical problems, or training, which was recognized in software maintenance ontology proposed by Kitchenham et al. [15]. In industrial practice, each software organization identifies and use typology of maintenance tasks that is the most suitable for its technical and organizational context.

Figure 2 presents a typical workload distributed to the common maintenance types during the phase of software product operational use after delivery. Faults are usually identified after software delivery, and the majority of them are solved as soon as it is possible. Later, faults are rarely identified during software use, resulting in a decreasing curve representing corrective maintenance task (red curve in figure 2). The numbers of other types of software maintenance tasks slightly increase leading to improvements, enhancements or adaptation of used software (other coloured lines in figure 2). In later phases of software use, after the majority of faults are detected and corrected, dominate enhancement of software (green curve in figure 2) and support activities to software users (purple curve in figure 2).

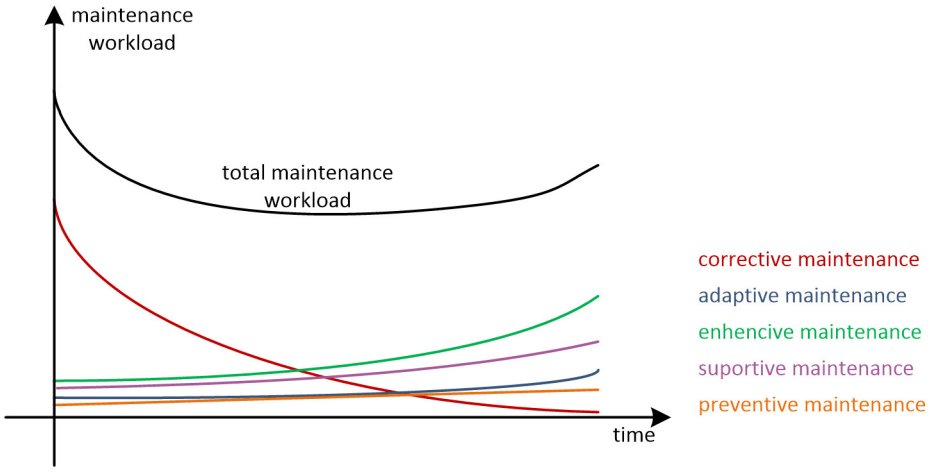

Figure 2. Typical maintenance workload after software delivery.

Different types of information that provide unique view on the specific maintenance context and tasks are commonly extracted from Software Historical Repositories (SHR) [16]. These repositories contain data about versions, bugs, communication records and other information about software products, tasks and projects, and clients. In addition, event logs that contain information on process execution are essential for assessing and improving software maintenance processes [17]. Each software organization records data specific for its practice, including the most relevant technical and organizational data. Based on the literature survey, Kagdi et al. [18] identified the following uses of repository data in the context of software evolution and maintenance: evolutionary couplings and patterns, change classification and representation, change comprehension, defect classification and analysis, source code differencing, origin analysis and refactoring, software reuse, development process and communication, contribution analysis, and evolution metrics. The basic idea with the use of SHR in software maintenance is to increase

maintainers efficiency by providing them the relevant data, which should reduce their reliance on their intuition and experience. In addition, historical data on maintenance everyday practice can be used to support management activities within an organization [19, 20, 21], or as a source of valuable data for maintenance practice assessment and improvement [22, 23].

3. Process assessment and improvement in small software companies

Small and very small software organizations are the global force of software industry, with tremendous impact on its competitiveness and innovation [24, 25, 26]. According to European Commission [27], Small and Medium-sized Enterprises (SMEs) comprise about 90% of all enterprises, providing about two-thirds of the jobs. Laporte et al. [28] reported that in Europe 85% of the companies in IT have only one to 10 employees, while in the Montreal area in Canada 50% of software development companies have fewer than 10 employees. L'Erario et al. reported that software SMEs in Brasil represent over 95% of all software organizations. Hasan et al. [29] stressed the importance of small software companies in the economies of many countries, including USA, China, Brazil, and European countries. These small companies have been recognized in industry as Very Small Entities (VSEs), which include enterprises, organizations, departments or teams having up to 25 people.

Practice improvement in software industry has been commonly implemented through Software Process Improvement (SPI) initiatives. These initiatives are supported by reference models and international standards, such as Capability Maturity Model Integration (CMMI) [30], and ISO/IEC 33001:2015 as a replacement for ISO/IEC 15504 (*Software Process Improvement and Capability Determination, SPICE*) [31], which adopt and implement large software organizations [32]. Chevers et al. [33] suggested that reference models should be simplified, more context relevant, and less disruptive for implementation in small organizations. Based on an empirical study Sharma and Sangal [26] identified 16 inhibitors for implementing SPI initiatives in software SMEs, among which the most important are the lack of management commitment, the lack of resources, and the lack of communication and information sharing. VSEs face the following barriers to implement SPI initiatives [34]:

- *Financial.* SPI implementation costs include employee effort, tool support, external consultancy for implementing international standards, accreditation and certification. VSEs should see quick return on investment in SPI, with clear and measurable economical results in order to accept SPI.
- *Business continuity.* These companies have small teams dedicated to everyday tasks, and implementation of SPI standards would negatively affect their work and overall business continuity.
- *Skilled experts.* The employees are experienced in the domain of work, while implementation of SPI requires specialized skills they do not have. This situation requires either attending certified courses for SPI experts, or hiring external experts, which VSEs usually cannot afford.
- *Organizational culture.* SPI initiative has significant impact on internal organization and organizational culture, which requires readiness for change and continuous improvement.
- *Selection of reference model.* The selection of the most suitable reference model or standard that is applicable in VSE setting is not easy and straightforward task. In many cases, implementation of reference models require reduction of their complexity and adaptation to specific settings.

In order to overcome the constraints of small software organizations in implementing software process assessment, a series of international standards *ISO/IEC TR 29110 Systems and software engineering - Lifecycle profiles for Very Small Entities (VSEs)* was developed. This series of

standards can be applied to any phase in software lifecycle. The purpose of a standard *ISO/IEC TR29110-3-1:2015 Systems and software engineering - Lifecycle profiles for Very Small Entities (VSEs) - Part 3-1: Assessment guide*[35] is to assists VSEs in assessing their processes.

Nevertheless of existence of standards, small software organizations do not adopt them for assessing and managing their processes. A survey conducted among software organizations worldwide indicates that less than 18% of VSEs are certified, while 53% of larger companies adopt standards [28]. In some cases, small and medium organizations tailor existing standards or reference models to their needs, like Process Assessment Method for Small to Medium Enterprises (PAM-SMEs) based on CMMI, which was successfully implemented in three companies [36].

Literature review related to software process assessment revealed that lightweight process assessment methods, which are designed and tailored based on personal experience, are more suitable for VSEs [32]. These lightweight methods are inductive (bottom-up), focused on the most critical segments of the practice, and facilitate organizational learning within organizations that implement them [37]. According to Pettersson et al. [38], light weight software process assessment and improvement should be based on the organization's experience and knowledge, and supported by triangulation of data (use of multiple data sources).

The stated importance of maintenance practice for software companies, and the growing share of small companies in the software industry indicate that it is necessary to provide methods and guides for managing and improving maintenance processes in small software organizations. Hasan et al. [29] conducted an interpretive case study in order to inquire maintenance process practice in a small software organization that develops and maintains an university information system. Results of the case study indicate that maintenance practice was not based on any defined process model, but rather use informal ad-hoc processes, which is the main source of problems. Pino et al. [39] presented Agile_MANTEMA, an agile methodology for implementing maintenance processes in small companies. This methodology consider types of maintenance, service levels and capability levels in maintenance process assessment. Implementation in two small companies revealed that Agile_MANTEMA is suitable methodology for maintenance process improvement. Based on a systematic literature review, de Melo Fran et al. [40] noticed that there is a lack of guidelines for the measurement process related to software maintenance practice in micro and small companies. Multiple case studies conducted by L'Erario et al. [41] in small software organizations revealed that they are not aware of software maintenance standards, and are not interested in implementing standards for process assessment and improvement.

Due to presented observations of the current industrial and research trends in software maintenance practice in small software organizations, there is a need for more research on software maintenance practice improvement in small software organizations (companies), with the focus on their specificities and constraints.

4. Software maintenance improvement project

Software maintenance process improvement project has been implemented in an indigenous micro software company in Serbia. The project was planned as a part of the project "The development of software tools for business process analysis and improvement", which is funded by the Ministry of Education, Science and Technological Development, Republic of Serbia. The company manager and the author of this paper jointly prepared the software maintenance process improvement project during the author's research work on a Ph.D. dissertation related to methods for software maintenance improvement. Project preparation was done in the period from 2009 to 2010, while the implementation started in 2011. The selection of the maintenance processes was grounded in the observation of the company manager that the maintenance activities consume majority of working hours of programmers, and the author's expertise with software maintenance concepts, principles and practice.

According to European Commission, the company is classified as a micro enterprise [27], since

it has seven employees, six programmers and one technical secretary. The company services are oriented towards local clients in Serbia, which significantly impacts the company's business strategy and internal organization. The company has over 100 clients that use over 40 business software applications. The company develops business software solutions, for which it provides full support and maintenance. Software maintenance services include regular maintenance in accordance with legal regulations, fixing bugs and enhancements of existing software solutions. Inquiry of the daily task trends in the company for the period from May 2010 to November 2011 revealed that over 84% of tasks are maintenance tasks [42], while a trend analysis of the tasks for 19 months starting from February 2013 revealed that over 88% of the tasks relate to maintenance [43]. These trends confirm the observation of the company manager before staring the project and clearly points out the importance of assessing and improving software maintenance processes in the company.

The company provides the maintenance services for over 100 client organizations, of which over 30 have signed Service Level Agreement (SLA) for maintenance services. SLA is a common way for arranging maintenance services in software industry [44]. Financial and technical details of maintenance services are determined in SLAs. When processing Maintenance Requests (MRs) from clients, the programmers consider whether the client signed SLA, since the clients with SLAs have a higher priority for service delivery. If a client submits a MR that fits into the SLA, maintenance services will not be charged. Otherwise, maintenance services are charged. For clients without SLA, all maintenance services are charged.

Programmers are associated to software applications in a way that enables the most effective organization of tasks in the company. To each software application are assigned two or more programmers, and each programmer is responsible for multiple software applications. In this way, the chances for finding a programmer that is free to accept a MR are greater, and the MR will be solved faster. In many cases, clients know who is responsible for which application and call one of the responsible programmers to report a MR.

4.1. Software maintenance practice in the company

The main entities in maintenance processes in the company are MRs, which are submitted by the clients. MRs provides the basis for organizing maintenance work in the company. Based on a MR analysis, a programmer identifies the necessary activities and creates a task and an associated working order. In this way, each MR and associated task are recorded in the internal repository. Based on the existence of a SLA or working order, the company charges maintenance services. Time processing of MRs is presented in figure 3.

Processing of a MR starts with recording it into the internal software application (*Recording date*). Recording of a MR is a duty of a programmer who received it. During the recording of the MR, the proposal of the deadline (*Proposed deadline*) stated by the user is also entered. After that, based on programmers' assignments to software application, a MR is forwarded to one of the most suitable programmers (*Assignment date*), who accept to solve it (*Acceptance date*). When a programmer finishes with the task associated to the MR, he marks that in the internal application (*Completion date*). Based on the completed work, a programmer closes a working order (*Working order date*), which is the basis for charging maintenance service.

Working on a MR may include work in the company that is recorded as *Company working hours*, accessing the clients infrastructure via Internet that is recorded as *Internet working hours*, and working within the client's organization that is recorded as *Client side working hours*.

For an urgent MR, which should be solved immediately because of the negative impact on the client's business, the MR is solved without recording it into the application. After completing all tasks related to that MR, it is recorded into the application. This is in the line with the company business strategy to provide valuable and on-time services to its clients.

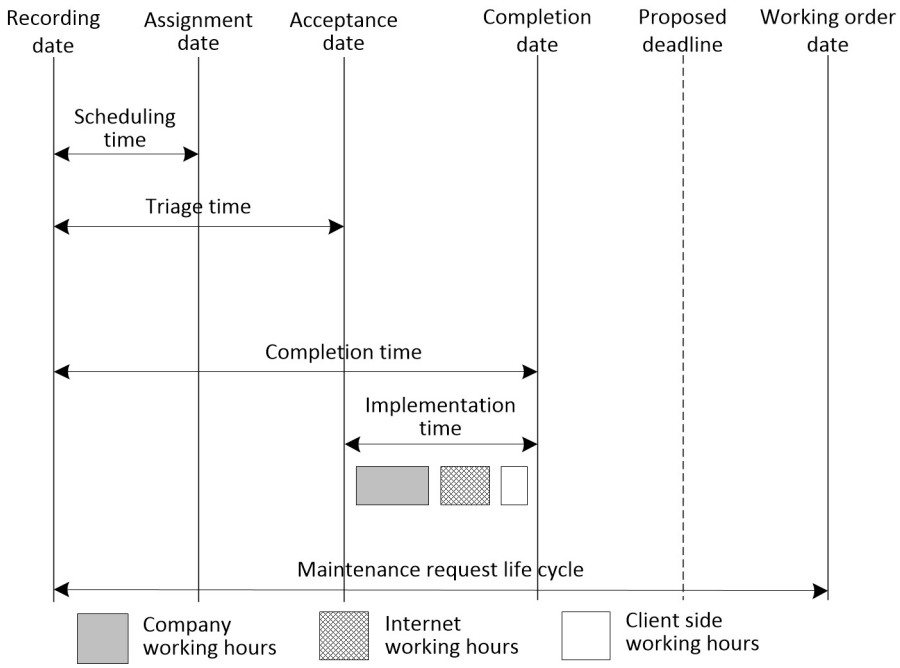

Figure 3. Maintenance request timeline.

4.2. Maintenance processes improvement
Since the planning and implementation of the process improvement require familiarization with the company's internal organization, it was necessary to spend few working days in the company before the planning phase. During these days, initial consultations were organized with the programmers and existing documentation was collected. Based on these consultations, an outline of the process improvement project is proposed by the company manager and the author of this paper, which is presented in figure 4.

Familiarization with the company internal organization includes getting initial insight into the everyday practice of all employees and into the typical processes. Access to technical infrastructure, the internal repository of tasks, and the documentation also helped in getting more detailed insight into the maintenance processes.

Improvement plan includes time frame for activities (process assessment, improvement implementation, tracking of results after improvement implementation), identification of necessary resources (human, technical, documentation), methods and tools to be used, and factors that can influence the project implementation. The critical success factors for implementing process improvement are [23]: company management support, availability of company employees, access to resources in the company, identification of processes in the company, and inclusion of additional researchers for specific research tasks.

Assessment plan includes time frame for doing process assessment (three to six months), a general plan for employee involvement, and selection of methods for collecting and analyzing data (both qualitative [45] and quantitative [46, 47] methods are identified as necessary). The main idea was to use lightweight approach that is flexible and can be adapted according to the situation in the company.

4.2.1. Maintenance process assessment Process assessment is the most critical and demanding phase in process improvement project since it requires long-term presence and monitoring of

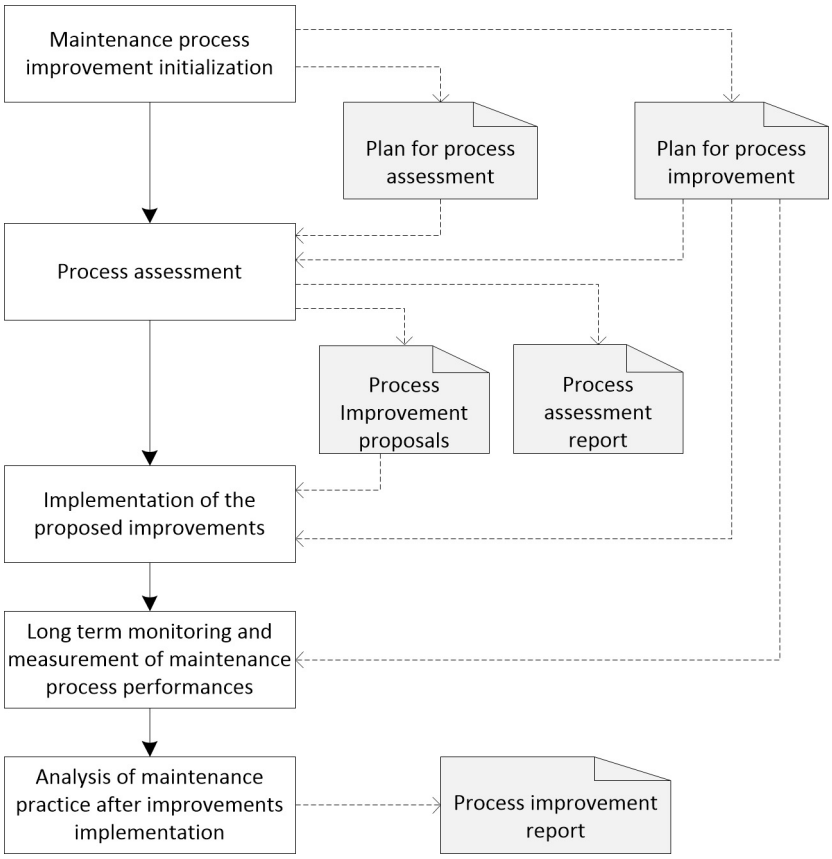

Figure 4. Maintenance process improvement project overview.

everyday working activities in the company. In addition, the company employees actively participate in the process assessment because they implement maintenance processes and are the best source of information about them. Developed process assessment method, named *Lightweight Inductive Method for Process Assessment based on Frequent Feedback (LIMPAF*^{2}*)* is presented in figure 5, while details are presented in [22].

The method is characterized as *inductive* because it starts with observing the real everyday practice in the company leading to the identified potential improvement grounded in the real needs. The main characteristics of the proposed assessment method are: (1) it is tailored to the real needs of an organization that assess processes, (2) it does not follow any prescribed framework/standard or best practice guidelines, (3) the method and an implementation study are prepared through joint work of researchers and organization staff, (4) an organization chooses what to assess and improve based on its real needs and business strategies, (5) the assessment focus is on the most critical aspects of the practice, (6) it facilitates organizational learning and knowledge sharing during the assessment process, and (7) it is suitable for small organizations. These characteristics enables implementation of the method in the selected micro software company.

The main elements of the proposed inductive process assessment method are [37]:

- *Inductive reasoning* is based on generalization of specific observations and experiences from the field work that leads to more general conclusions and results [48].
- *Triangulation of data sources and methods* for analyzing data assumes the use of different data sources within an organization, which includes both quantitative and qualitative data

sources [49, 50].

- *Regular feedback* to organization during the whole assessment process, which ensures that all assessment activities are properly implemented and assessment results are validated by the company management [51, 52].
- *Support for organizational learning* in the organization that assess processes through identification and systematization of knowledge that exists in the company [53, 54, 55].

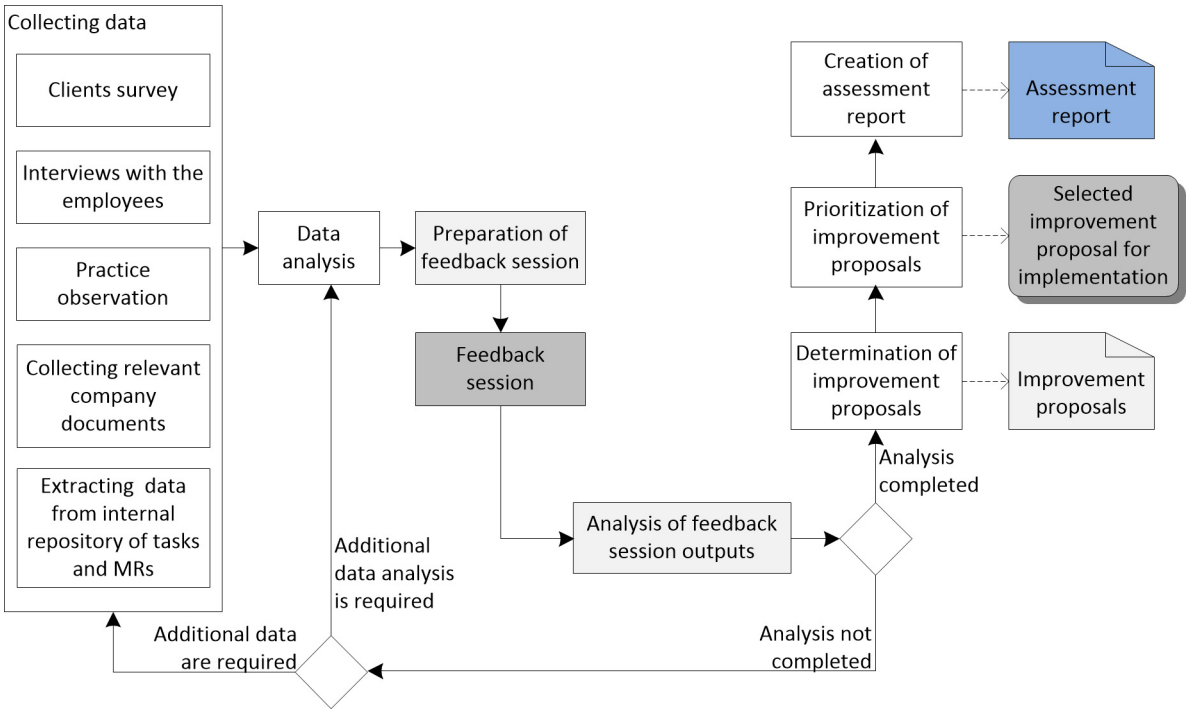

Figure 5. Maintenance process assessment overview.

The development of the *lightweight* assessment method was driven by the following objectives: (1) quick and inexpensive process assessment suitable for small companies that do not have resources to implement heavyweight and expensive standard based approaches, (2) easy diagnosis of the selected processes and proposal of potential improvements, (3) to allow engineers to work on their daily tasks with low engagement in the assessment process that do not disturb their daily routines, and (4) frequent feedback sessions enable practice assessment and adjustment of improvement proposals through joint work of employees and researchers.

The basic data analysis technique in the process assessment method is inductive thematic analysis [56], which is used for exploring characteristics of maintenance processes [57]. Several quantitative data analysis methods were used for accompanying and justifying qualitative findings, such as trend analysis [43, 58], regression analysis for effort estimation [59, 60] and task complexity analysis [61], and fuzzy screening for process evaluation [62]. This complex set of quantitative and qualitative data analysis techniques supports triangulation of used data sources and increases the validity of the findings. The main findings of data analysis are:

- *Prioritized improvement proposals.* A set of potential improvements is identified through inductive thematic analysis of interviews with employees and records of feedback sessions, while the ranking of the improvements was done based on expert opinions by using fuzzy

screening method. In this way, the relevance of the proposed improvements was judged based on their criticality for the company maintenance practice.

- *Descriptions of processes' capabilities and features.* The documents with detailed descriptions of processes (text documents, tables and graphs) are based on the data obtained after completing all necessary analysis during the process assessment.
- *Systematized knowledge about maintenance practice.* Process assessment with detailed insight into the everyday practice is used for identification and systematization of knowledge relevant for maintenance practice in the company. The use of inductive thematic analysis ensures that the systematized knowledge reflects existing knowledge in the company.

4.2.2. Improvement implementation After completing process assessment phase, the following proposals for maintenance process improvement were identified:

- Optimization of maintenance requests' processing timeline with the focus on recording and triaging maintenance requests.
- Optimization of maintenance tasks scheduling based on evidence of working hours distribution per maintenance requests.
- Development of a web based application for reporting maintenance requests, which will be available to users and integrated with the internal software application for tracking the tasks in the company.
- Development of a software solution for reporting on maintenance request processing based on quantitative analysis of data recorded in the internal repository of programmers' tasks.

Programmers ranked identified improvement proposals by using fuzzy screening method [47], based on the most relevant criteria for maintenance practice in the company (request scheduling time, request acceptance time, request completion time, and working hours spent on requests). For the implementation was selected the first improvement proposal related to maintenance request timeline optimization.

The selected improvement refers to the introduction of a new time determinant (*Start date*) in the MR process timeline that provides more accurate monitoring of the process of request realization and time spent by the programmer for solving the request. In this way, better scheduling of MRs and business management is expected, which leads to higher earnings in software maintenance. The previous MR processing timeline presented in figure 3 is imprecise in terms of tracking the exact start of work on a MR and time spent for solving it. Actually, in many cases programmers postpone a MR implementation due to current work, and *Acceptance date* usually do not reflect the exact date of staring work on the MR. New MR processsing timeline, with introduction of the exact date of starting work on a MR (*Start date*) is presented in figure 6. In this way, the implementation time of MR is more accurate and enable better management of maintenance activities based on the actual programmers workload.

The implementation of a selected improvement includes a modification of the internal software for tracking maintenance requests and programmers' tasks. The improvement implementation required the modification of the logic layer and database layer in the internal software application. The improvement was implemented as a technical solution in 2013. and regularly has been used in the company's everyday practice.

4.3. Organizational learning and maintenance knowledge systematization

Software process improvement initiatives enable and support organizational learning activities within the organization that assess and improve processes [53, 63, 64]. Active participation of the company employees in all improvement activities ensures identification of relevant knowledge about the current practice, leading to identification of the most relevant improvement proposals.

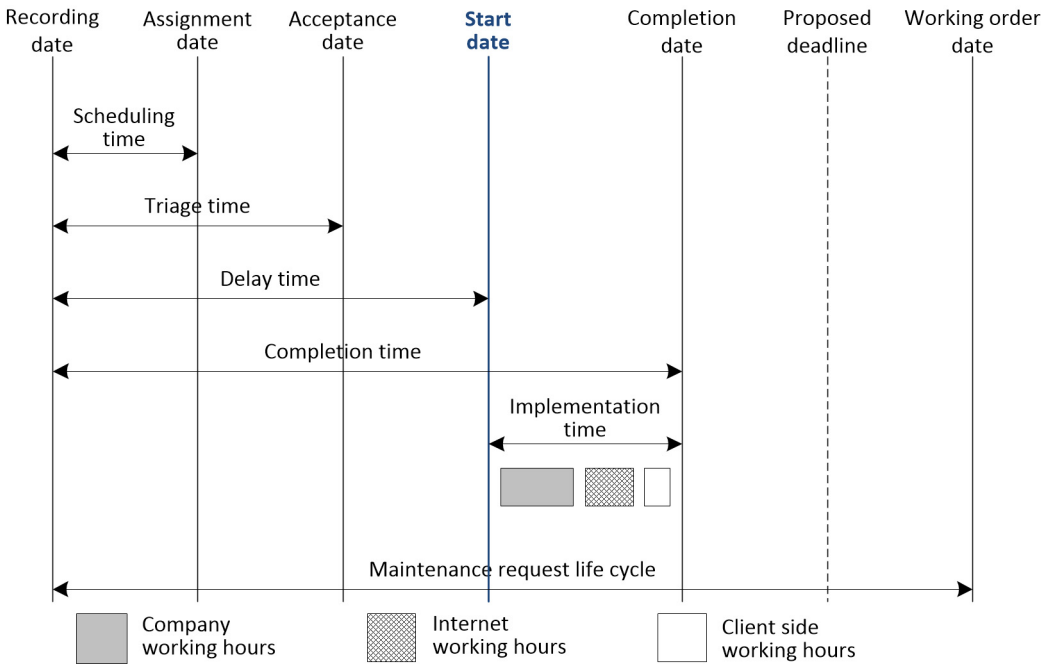

Figure 6. Improved maintenance request timeline.

It has been recognized that refinement and reuse of existing knowledge in software companies support practice improvement and increase the quality of services provided to customers [65, 66]. These observations and large amount of data collected during maintenance process assessment, motivated initiative for identifying and structuring knowledge on software maintenance practice.

Since the knowledge about the practice and processes resides in the employees minds, the focus during the process assessment was on interviewing them, observing them during everyday work activities and including them in data analysis, especially through feedback sessions organized in the company. In this way, majority of the data collected during process assessment activities can be used as a basis for knowledge identification. Critical success factors for implementing knowledge identification and systematization are [67]: full support of the company management, availability of the company employees based on the project needs, motivating the employees to actively participate in the project, access to all resources in the company, access to the company clients, implementing research activities in a way that does not disrupt everyday activities of the employees, inclusion of the company employees in the data analysis, validation of the project findings by the company management, and including external researchers based on the project needs and their experience with quantitative data analysis methods.

The method for identification and systematization of knowledge in micro software companies (teams) was create in order to provide a framework for knowledge management activities in the context of software process improvement projects [55]. The method *Lightweight Inductive Method for Knowledge Identification and Systematization (LIM4KIS)* was designed through joint work of the author of this paper and the company manager of the company in which process improvement was implemented [21]. The *LIM4KIS* is designed as lightweight (low level of employees engagement and use of available resources in the company) and inductive (starts with everyday practice and organizational context in the company), which enables incremental development of knowledge framework related to maintenance practice in the company. Knowledge identification and systematization activities during software process

improvement project are presented in figure 7.

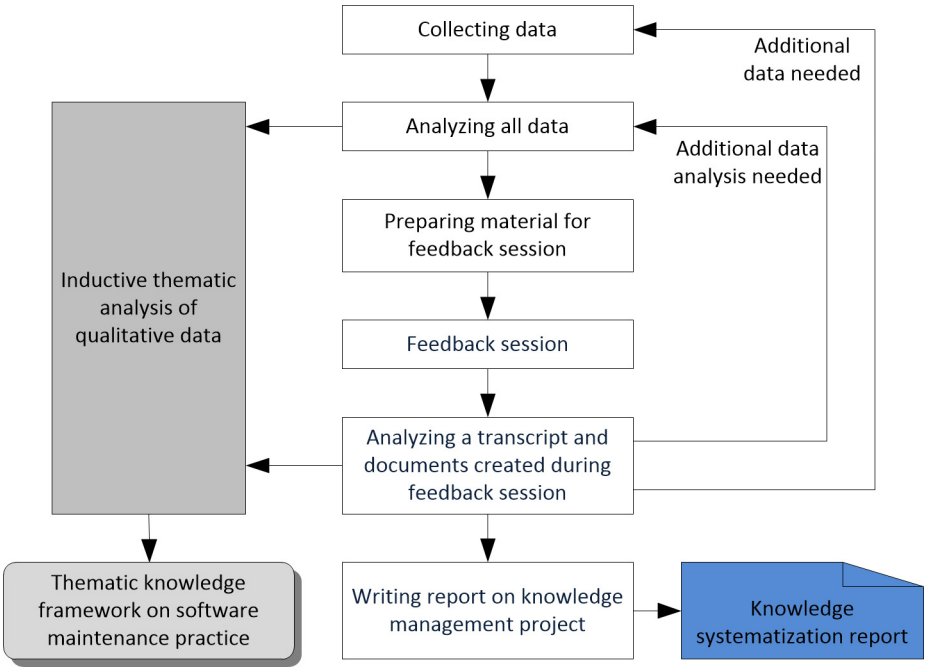

Figure 7. Overview of knowledge identification and systematization activities within software process improvement project.

By using inductive thematic analysis [56], a method suitable for identifying themes in unstructured text, a thematic knowledge framework with themes and sub-themes related to maintenance practice was developed. The basic activities in inductive thematic data analysis are: familiarization with the data, initial coding of unstructured text, theme development, and structuring identified themes in an hierarchical thematic framework. Hierarchical organization of themes and sub-themes enable fine-grained organization of knowledge about the company organizational context, employees and their working activities. The themes are organized in three thematic areas, representing areas that influence software maintenance work in the company:

- *Business policies and organizational issues.* Maintenance activities are aligned with the company internal organization and business strategy, which includes managing software products, programmers work, and customer relationships.
- *Human factor.* This area contains themes related to characteristics of *programmers* and *software users* as the main actors in software maintenance activities. Complexity of maintenance tasks requires programmers with cognitive skills (awareness of problem complexity, reasoning), organizational skills (self-organization, team work), and experience (problem recognition and association with the previous solved problems).
- *Processing maintenance requests.* This is the main thematic area that includes characteristics of the maintenance processes, and characteristics of maintenance requests as sources of work in maintenance.

Implementation of knowledge management activities through identification and systematization of software maintenance related knowledge brings the following benefits for the company:

- Creation of knowledge base on software maintenance practice that can be easily updated as maintenance practice evolve in the company. This knowledge base is especially important for young programmers that meet many problems for the first time.
- Increased employees satisfaction and the sense of the personal importance in the company because of active participation in the research project which outputs contribute to the better organization of work in the company.
- Adoption of knowledge management practice and culture in the company, which is essential for continuous practice observation and improvement.

5. Conclusions

This paper presents the author's reflections on personal experience related to inquiring and improving software maintenance practice in a local micro software company. Since the selected software company spends majority of working time on maintenance activities, assessment and improvement of software maintenance processes is essential for achieving better business performance and increased customer satisfaction. One of the identified improvement proposals was implemented as a technical solution in the company, which improved the time processing of maintenance requests. Process assessment and improvement facilitated systematization of knowledge on maintenance practice, which is additional benefit for the company.

Further work includes assessment and improvement of other business processes in the selected company, such as requirements engineering or reengineering of legacy systems, and adaptation of the developed process assessment method for implementation in other similar micro and small companies.

Acknowledgment

Ministry of Education, Science and Technological Development, Republic of Serbia, supports this research under the project "The development of software tools for business process analysis and improvement", project number TR32044.

References

[1] Sommerville I 2011 *Software Engineering* 9th ed (Boston, MA, USA: Addison Wesley)
[2] Bourque P and Fairley R E D (eds) 2014 *Guide to the Software Engineering Body of Knowledge (SWEBOK), Guide V3.0* 2014th ed (Piscataway, NJ, USA: IEEE Press)
[3] Ulziit B, Warraich Z A, Gencel C and Petersen K 2015 *Journal of Software: Evolution and Process* **27** 763–792 doi: 10.1002/smr.1720
[4] Juergens E 2016 *it - Information Technology* **58** 145–149 doi: 10.1515/itit-2016-0014
[5] Lenarduzzi V, Sillitti A and Taibi D 2017 *Proceddings of 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)* (Buenos Aires, Argentina) pp 146–148 doi: 10.1109/ICSE-C.2017.122
[6] Stojanov Z, Dobrilovic D and Stojanov J 2018 *Enterprise Information Systems* **12** 982–1006 doi: 10.1080/17517575.2018.1445296
[7] April A, Huffman Hayes J, Abran A and Dumke R 2005 *Journal of Software Maintenance and Evolution: Research and Practice* **17** 197–223 doi: 10.1002/smr.311
[8] ISO/IEEE 2006 *International Standard - ISO/IEC 14764 IEEE Std 14764-2006. Software Engineering - Software Life Cycle Processes - Maintenance* 2nd ed (Piscataway, NJ, USA: ISO)
[9] Tripathy P and Naik K 2015 *Software evolution and maintenance: a practitioner' s approach* (Hoboken, New Jersey, USA: John Wiley & Sons) doi: 10.1002/9781118964637
[10] Haworth D A, Sharpe S and Hale D P 1992 *Journal of Software Maintenance: Research and Practice* **4** 105–117 doi: 10.1002/smr.4360040204
[11] Grubb P and Takang A A 2003 *Software Maintenance: Concepts and Practice* 2nd ed (Singapore: World Scientific Publishing Company)
[12] Tsunoda M, Monden A, Matsumoto K, Ohiwa S and Oshino T 2021 *IEICE TRANSACTIONS on Information and Systems* **E104-D** 76–90 doi: 10.1587/transinf.2020MPP0004

[13] Swanson E B 1976 *Proceedings of the 2nd international conference on Software engineering* ICSE '76 (San Francisco, CA, USA) pp 492–497
[14] Chapin N 2000 *Proceedings of the International Conference on Software Maintenance* ICSM '00 (San Jose, CA, USA) pp 247– doi: 10.1109/ICSM.2000.883056
[15] Kitchenham B A, Travassos G H, von Mayrhauser A, Niessink F, Schneidewind N F, Singer J, Takada S, Vehvilainen R and Yang H 1999 *Journal of Software Maintenance: Research and Practice* **11** 365–389
[16] Sun X, Li B, Li Y and Chen Y 2015 *Computer and Information Science* (*Studies in Computational Intelligence* vol 566) ed Lee R (Springer, Cham) pp 27–37 doi: 10.1007/978-3-319-10509-3_3
[17] Gupta M, Serebrenik A and Jalote P 2017 *Proceedings of 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)* (Shanghai, China) pp 681–686 doi: 10.1109/ICSME.2017.39
[18] Kagdi H, Collard M L and Maletic J I 2007 *Journal of Software Maintenance and Evolution: Research and Practice* **19**(2) 77–131 doi: 10.1002/smr.344
[19] Banker R D, Datar S M, Kemerer C F and Zweig D 2002 *Information Technology and Management* **3**(1-2) 25–41 doi: 10.1023/A:1013156608583
[20] April A 2010 *Proceedings of the Seventh International Conference on the Quality of Information and Communications Technology* (Porto, Portugal) pp 352–357 doi: 10.1109/QUATIC.2010.65
[21] Stojanov Z 2019 *Journal of Software Engineering & Intelligent Systems* **4** 41–57
[22] Stojanov Z, Stojanov J and Dobrilovic D 2019 *Journal of Engineering Management and Competitiveness* **9** 134–147
[23] Stojanov Z 2020 *Proceedings of the 10th International conference on Applied Internet and Information Technologies (AIIT2020)* (Zrenjanin, Serbia) pp 141–146
[24] Richardson I and von Wangenheim C G 2007 *IEEE Software* **24** 18–22 ISSN 0740-7459
[25] Giardino C, Unterkalmsteiner M, Paternoster N, Gorschek T and Abrahamsson P 2014 *IEEE Software* **31** 28–32 doi: 10.1109/MS.2014.129
[26] Sharma P and Sangal A L 2018 *Journal of Software: Evolution and Process* **30** e1993 doi: 10.1002/smr.1993
[27] European Commission 2015 *User guide to the SME Definition* Enterprise and industry publications (Luxembourg: Publications Office of the European Union)
[28] Laporte C Y, Alexandre S and Renault A 2008 *Computer* **41**(3) 98–101 doi: 10.1109/MC.2008.86
[29] Hasan R, Chakraborty S and Dehlinger J 2012 *Software Engineering Research,Management and Applications 2011* (*Studies in Computational Intelligence* vol 377) ed Lee R (Springer, Berlin, Heidelberg) pp 129–143 doi: 10.1007/978-3-642-23202-2_9
[30] CMMI Product Team 2010 *CMMI for Development, Version 1.3. Improving processes for developing better products and services* Technical Report CMU/SEI-2010-TR-033 (Pittsburgh, PA, USA: Software Engineering Institute, Carnegie Mellon University)
[31] ISO/IEC 2015 *Interantional standard ISO/IEC 33001:2015 Information technology — Process assessment — Concepts and terminology* (Geneva, Switzerland: International Organization for Standardization)
[32] Zarour M, Abran A, Desharnais J M and Alarifi A 2015 *Journal of Systems and Software* **101** 180–192 doi: 10.1016/j.jss.2014.11.041
[33] Chevers D A, Mills A M, Duggan E W and Moore S E 2017 *Journal of Global Information Technology Management* **20** 110–130 doi: 10.1080/1097198X.2017.1321356
[34] Larrucea X, O'Connor R V, Colomo-Palacios R and Laporte C Y 2016 *IEEE Software* **33** 85–89 doi: 10.1109/MS.2016.42
[35] ISO/IEC 2015 *Interantional standard ISO/IEC TR29110-3-1:2015 Systems and software engineering - Lifecycle profiles for Very Small Entities (VSEs) - Part 3-1: Assessment guide* (Geneva, Switzerland: International Organization for Standardization)
[36] Abushama H M 2016 *Journal of Software: Evolution and Process* **28** 689–711 doi: 10.1002/smr.1793
[37] Stojanov Z 2016 *Proceedings of the 6th International conference on Applied Internet and Information Technologies (AIIT2016)* (Bitola, North Macedonia) pp I–XV doi: 10.20544/AIIT2016.I01 [plenary speach]
[38] Pettersson F, Ivarsson M, Gorschek T and Öhman P 2008 *Journal of Systems and Software* **81** 972–995 doi: 10.1016/j.jss.2007.08.032
[39] Pino F J, Ruiz F, García F and Piattini M 2012 *Journal of Software: Evolution and Process* **24** 851–876 ISSN 1532-0618 dOI: 10.1002/smr.541 URL `http://dx.doi.org/10.1002/smr.541`
[40] de Melo V J A T, Leal G C L, Balancieri R, Rouiller A C *et al.* 2020 *Independent Journal of Management & Production* **11** 519–537 doi: 10.14807/ijmp.v11i2.1028
[41] L'Erario A, Thomazinho H C S and Fabri J A 2020 *International Journal of Software Engineering and Knowledge Engineering* **30** 603–630 doi:10.1142/S0218194020500217
[42] Stojanov Z, Dobrilovic D and Stojanov J 2013 *Theory and Applications of Mathematics & Computer Science* **3** 59–74
[43] Stojanov Z, Stojanov J, Dobrilovic D and Petrov N 2017 *Proceedings of the IEEE 15th International*

Symposium on Intelligent Systems and Informatics (SISY 2017) (Subotica, Serbia) pp 23–27 doi: 10.1109/SISY.2017.8080547
[44] April A and Abran A 2008 *Software Maintenance Management: Evaluation and Continuous Improvement* (Hoboken, NJ, USA: Wiley-IEEE Computer Society)
[45] Stojanov Z 2015 *Encyclopedia of Information Science and Technology* ed Khosrow-Pour M (Hershey, PA, USA: IGI Global) chap 62, pp 650–658 3rd ed doi: 10.4018/978-1-4666-5888-2.ch062
[46] Buglear J 2001 *Stats means business: a guide to business statistics* (Oxford, UK: Buttworth-Heinemann)
[47] Yager R R 1993 *Fuzzy Logic* (*Theory and Decision Library (Series D: System Theory, Knowledge Engineering and Problem Solving)* vol 12) ed Lowen R and Roubens M (Springer, Dordrecht) pp 251–261 doi: 10.1007/978-94-011-2014-2_24
[48] Molnar G, Greiff S and Csapo B 2013 *Thinking Skills and Creativity* **9** 35–45 doi: 10.1016/j.tsc.2013.03.002
[49] Lethbridge T C, Sim S E and Singer J 2005 *Empirical Software Engineering* **10** 311–341 doi: 10.1007/s10664-005-1290-x
[50] Miller J 2008 *Empirical Software Engineering* **13**(2) 223–228 doi: 10.1007/s10664-008-9063-y
[51] Roebuck C 1996 *Long Range Planning* **29** 328–336
[52] Hattie J and Timperley H 2007 *Review of Educational Research* **77** 81–112
[53] Arent J, Iversen J H, Andersen C V and Bang S 2000 *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences* (Maui, Hawaii, USA) pp 10 pp.–
[54] Argote L 2013 *Organizational Learning: Creating, Retaining and Transferring Knowledge* 2nd ed (New York, USA: Springer US)
[55] Stojanov Z, Stojanov J and Dobrilovic D 2015 *Proceedings of the IEEE 13th International Symposium on Intelligent Systems and Informatics (SISY 2015)* (Subotica, Serbia) pp 25–30 doi: 10.1109/SISY.2015.7325405
[56] Braun V and Clarke V 2006 *Qualitative Research in Psychology* **3** 77–101 doi: 10.1191/1478088706qp063oa
[57] Stojanov Z and Stojanov J 2016 *Proceedings of the 6th International conference on Applied Internet and Information Technologies (AIIT2016)* (Bitola, North Macedonia) pp 9–17 doi: 10.20544/AIIT2016.02
[58] Stojanov Z and Stojanov J 2018 *Proceedings of the 8th International conference on Applied Internet and Information Technologies (AIIT2018)* (Bitola, North Macedonia) pp 73–77 doi: 10.20544/AIIT2018.P16
[59] Stojanov Z, Dobrilovic D, Stojanov J and Jevtic V 2013 *Scientific Bulletin of The "Politehnica" University of Timisoara, Transactions on Automatic Control and Computer Science* **58** 131–138
[60] Stojanov Z, Dobrilovic D, Stojanov J and Jevtic V 2013 *Proceedings of the IEEE 8th International Symposium on Applied Computational Intelligence and Informatics (SACI 2013)* (Timisoara, Romania) pp 461–466 doi: 10.1109/SACI.2013.6609019
[61] Stojanov Z, Stojanov J and Dobrilovic D 2018 *Theory and Applications of Mathematics & Computer Science* **8** 24–38
[62] Stojanov Z, Brtka V and Dobrilovic D 2014 *Proceedings of the IEEE 9th International Symposium on Applied Computational Intelligence and Informatics (SACI 2014)* (Timisoara, Romania) pp 67–72 doi: 10.1109/SACI.2014.6840037
[63] Dybå T, Dingsøyr T and Moe N B 2004 *Process Improvement in Practice - A Handbook for IT Companies* (*International Series in Software Engineering* vol 9) (Norwell, MA, USA: Kluwer Academic Publishers)
[64] Menolli A L A, Reinehr S and Malucelli A 2013 *International Journal of Software Engineering and Knowledge Engineering* **23** 1153–1175 doi:10.1142/S0218194013500356
[65] Alagarsamy K, Justus S and Iyakutti K 2007 *International Conference on Software Engineering Advances (ICSEA 2007)* (Cap Esterel, France) pp 61–67 doi: 10.1109/ICSEA.2007.73
[66] Ivarsson M and Gorschek T 2012 *Software Quality Journal* **20** 173–199 doi: 10.1007/s11219-011-9139-6
[67] Stojanov Z and Dobrilovic D 2019 *Proceedings of the 11th International conference Digital transformation of the economy and society: shaping the future* (Prilep, North Macedonia) pp 284–292