Theory and Applications of Mathematics & Computer Science

(ISSN 2067-2764, EISSN 2247-6202)
http://www.uav.ro/applications/se/journal/index.php/tamcs

Theory and Applications of Mathematics & Computer Science 3 (2) (2013) 59–74

Analyzing Trends for Maintenance Request Process Assessment: Empirical Investigation in a Very Small Software Company

Zeljko Stojanov[a,*], Dalibor Dobrilovic[a], Jelena Stojanov[a]

[a]*University of Novi Sad, Technical Faculty "Mihajlo Pupin", Djure Djakovica BB, 23000 Zrenjanin, Serbia*

Abstract

Assessment and improvement of software maintenance processes in small software companies is very important because of large costs of maintenance and constraints of small software companies. This study presents an approach to assessment of software maintenance requests' processing in a very small local software company. The approach is context dependent and uses trend analysis and feedback sessions for assessing the current state of maintenance request processing. The analysis is based on various sources of data such as: internal repository of maintenance requests, company documents, transcribed records of interviews with company employees, and transcribed records of feedback sessions. Monthly trends for maintenance requests, working hours and types of maintenance tasks, by considering clients and software products are presented in the article. Identified trends were discussed during feedback sessions in the company. Participants in feedback sessions were company employees and researchers. During discussions of trends, some directions for further improvement of maintenance requests' processing were proposed. The article concludes with implications for practitioners from industry and researchers, as well as further research directions.

Keywords: software maintenance, process assessment, trend analysis, feedback session, very small software company.
ACM CCS: D.2.7 Distribution, Maintenance, and Enhancement, D.2.9 Management—Life cycle, K.6.3 Software Management—Software maintenance.

1. Introduction

Software maintenance includes all activities related to the preservation of consistency and efficiency of complex software systems. Maintenance consumes most of the costs of software systems (between 40% and 90% of the total costs) in software life-cycle (Lientz *et al.*, 1978; Kajko-Mattsson *et al.*, 2001; Abran *et al.*, 2004). Maintenance costs for systems that are in use for a very long time usually greatly exceed the costs of development. Despite that fact, software maintenance attracts less attention in scientific literature comparing to software development.

*Corresponding author
Email addresses: zeljko.stojanov@tfzr.rs (Zeljko Stojanov), ddobrilo@tfzr.rs (Dalibor Dobrilovic), jelena@tfzr.uns.ac.rs (Jelena Stojanov)

Software Maintenance is in literature recognized as the last phase in software life-cycle, which does not attract enough attention when compared with software development. Developers and managers consider maintenance requests as short-term jobs that should be done as quickly as it is possible (Junio *et al.*, 2011). Research on the maintenance process conducted with people involved in the process indicates that only 2.7% thought that the maintenance process is effective, while 70.2% of them believe that the maintenance process is ineffective (Sousa, 1998).

Small companies are dominant in economies across the globe (Richardson & von Wangenheim, 2007). U.S. Census Bureaus "1995 County Business Patterns" pointed that the vast majority of software and data processing companies are small, and that those with more than 50 employees comprise only a few percent of the total number (Fayad *et al.*, 2000). Laporte *et al.* (2006) reported that in Europe, 85% of IT sector companies have between 1 and 10 employees.

Small software companies are typically characterized as economically vulnerable with low budget to perform corrective post delivery maintenance, as well as with limited resources and the lack of knowledge and capacities to implement software process assessment and improvement activities (Laporte *et al.*, 2008). According to Vasilev (2012), rationalization of processes indirectly affects company business and reduces managerial costs. Small software companies have not adopted assessment directives proposed by software process improvement (SPI) models (Capability Maturity Model (CMM) and more recently CMMI) or intenational process-related standards (ISO 15504 and ISO 9001) (Fayad & Laitinen, 1997). Because of limitations in scale and resources, small software companies find software process improvement a major challenge that should be supported with short, light and tailored assessment methods (Mc Caffery *et al.*, 2007). Qualitative empirical study about the maintenance practice in local small software companies (companies from Timisoara and Zrenjanin), with the focus on collecting and processing client requests, revealed that they face many problems, both technical and organizational (Stojanov, 2011; Stojanov *et al.*, 2011; Stojanov, 2012*b*). Therefore, software maintenance practice assessment and improvement in these companies require more attention.

This paper presents an approach to maintenance assessment in a very small software company based on analyzing trends in available maintenance data. Practice assessment is based on a tailored lightweight approach with frequent feedback, with the following phases: initialization, planning, execution, and final reporting on assessment. Since the aim of this paper is to present the use of trend analysis as a valuable tool in process assessment, assessment phases will not be discussed in more details.

The research was conducted in a very small software company with seven employees (classified as micro enterprise according to (Commission, 2005)). This study is a part of a large project (from 2011 to 2014) with the aim to assess and improve maintenance practice in the selected software company. Data collected in the company through practice observation, interviews with programmers, and analysis of company documents and maintenance repository provide the basis for assessment of the maintenance practice. The objective of the paper is to present a light assessment approach of software maintenance practice based on trend analysis.

The paper is organized as follows. Section 2 contains related work that presents the use of trend analysis in software engineering. After that are described the context of the research in section 3, and analysis of maintenance trends in section 4. Discussion of results follows in section 5, while discussion of treats to validity is in section 6. The last section of the paper contains concluding

remarks with implications for research community and practitioners from industry, and further research directions.

2. Related work

Software maintenance trend analysis requires systematic data collection over an appropriate period. This is very important since maintenance requests occur randomly, and cannot be planned neither technically nor in budget. It is also important to note that maintenance workload cannot be managed using project management techniques (April, 2010).

Trend analysis can help in analyzing and controlling the activities and processes, and in assessing the efficiency of observed processes. Trend analysis is based on real empirical data and provides information of prime importance for organization (Buglear, 2001). A trend can be seen as an underlying longer-term movement in the observed data series. In addition, trend analysis is also important for providing evidence on deviations from trends. Trends are generally related to long-term observation and data collection, although the term "*long term*" is based on the subjective assessment (Chatfield, 1996). Kanoun & Laprie (1996) argued that trend analyses are usually applied intuitively and empirically rather than in quantified and well-defined manner. Results of trend analyses provide valuable information for assessing maintenance activities and workload of maintenance personnel.

In the paper (Kenmei *et al.*, 2008) is proposed a trend analysis approach of change requests based on time series analysis of data extracted from version control and bug tracking systems. The empirical study is based on data from three large-scale open source software projects (Eclipse, Mozilla and JBoss). The study proved that time series are efficient tools for modeling change request density and further trends in receiving new change requests.

The study (Ahmed & Gokhale, 2009) presented an approach to modelling the behaviour of bugs inside Linux kernels. The study included the analysis of bug distribution, lifetime, and clustering inside the kernel modules, as well as a deeper analysis of the statistical trends in the bug data from an architectural perspective. The aim of the study was to gain insight into the factors that impact system reliability. The analysis was related to: trends across the three releases of the kernel, the manner in which bugs were resolved, and on understanding the impact of bug severities on the resolution time of the bugs. From the architectural perspective of the kernel, the results of the study based on the statistical trends suggested that the module dependencies and interactions have higher impact on the bugs than the individual modules themselves.

April (2010) presented trend analysis of software maintenance services. Analysis includes supply and demand of software maintenance services. The research was conducted as a part of a process improvement activity in Integratik, an ERP development firm in Canada. The improvement aim is the implementation of maintenance request tracking process and information system. This process should ensure that each request would be recorded, dispatched and tracked formally, as well as time recording of maintenance personnel effort. In addition, this improvement ensured that the maintenance demands would be properly measured and analyzed. The author investigated trends related to distribution of requests per months, maintenance personnel effort per months, distribution of requests and work effort for particular software applications.

Zhu *et al.* (2011) proposed an approach for quality evolution monitoring based on the analysis of deviation trends of different modularity views of software. The approach includes monitoring the following views: package view that prescribes how developers intentionally group related source files as modules (packages), structural cluster view that reflects the nature of inter-file dependencies and method invocation relations, and semantic cluster view that reflects the nature of vocabulary used and topics involved in different source files and their correlations. The approach is based on an assumption that the deviation between different modularity views can influence quality evolution. If the views are properly aligned, the developers will be able to easily locate concepts and implement modifications. Deviation between different trends was measured with SiMo (Similarity between the Modularity views) metrics. Deviations for individual versions were computed and analyzed, and after that deviation trends in a sequence of versions were analyzed. The main activities in the approach are: construction of modularity views, computation of similarity metrics and analysis of deviation trends. Deviation trend of different modularity views is a combination of three change trends (i.e. rise, drop, hold) of three SiMo metrics, which is totally 27 patterns of deviation trends. Empirical study was conducted on three open-source Java systems, JFreeChart, JHotDraw and Jedit, that are available at SourceForge.net Subversion (SVN) repositories. Presented empirical study confirms that continuous monitoring of deviation trends provides useful feedback.

3. Context

Proper understanding of the assessment approach requires more detailed insight into the organizational context where the study is conducted. The approach is tailored to a selected small software company and therefore it is necessary to outline basic facts about the company.

This research was realized in a very small software company with seven employees (six programmers and one technical secretary). Software development and maintenance activities are organized in the way that one or more programmers are assigned to each software application. When a maintenance request (MR) is received from a client, it is forwarded to a programmer from a set of assigned programmers. Programmers' assignments to software applications are documented and available to all employees.

The company maintains over 30 business software applications used by local clients in Serbia. Clients are classified in two groups: clients that have signed Maintenance Service Agreement (MSA) and pay for maintenance services on the monthly basis, and clients that do not have signed MSA and pay for each maintenance service after its completion.

3.1. Maintenance request processing

Analysis of trends for a long period of time requires the existence of systematically collected data, and the process that is implemented and followed by all stakeholders. All requests are recorded in the internal software application with the repository for issues tracking (requests, tasks, work orders). Although a process is usually tailored for the current request and a user, a general process is defined and implemented in the company. The process includes the following steps: receiving and recording a request in the internal repository, sending a notification email with the request info to an appropriate programmer, checking a client's status in order to define

the priority of the request, assigning a programmer to a request, collecting additional information that is necessary for understanding and solving a request (if necessary), preparing a bid for work that is supposed to be done for clients that do not have signed MSA, and solving the request.

The request processing is completed when a client confirms correctness of finished work. In the repository is recorded who confirms the correctness, the date and the way of confirming. After that a working order is printed and sent to a client, and a request is labeled as closed.

3.2. Internal repository

The repository provides the efficient platform for storing and tracking tasks and maintenance requests. Practically, maintenance requests include all types of requests for maintenance, not only requests related to modification of software applications. In order to provide support for complete set of activities related to tracking requests, in the repository are also stored data about clients, software applications and work orders that are associated to requests. The repository is managed through an internal Web based application.

Issue tracking system does not contain only records for clients maintenance requests, but also records for all other, non-maintenance, tasks. However, analysis of all records for the period from May 2010 to November 2011 provides the evidence that 1896 tasks of totally 2252 tasks are related to software maintenance (84%), while 356 tasks are related to other activities (16%). A period of 19 months that begins two years after introducing the issue tracking system in the company is selected for the analysis. Discussions with programmers in the company confirmed that they are accustomed in using the system, which ensures extraction of more reliable data from the repository.

3.3. Programmers' working hours

Working hours spent on solving clients' requests provide the real basis for charging maintenance services. These working hours are hours that a programmer spends on a specific task associated to a request. In addition, these working hours are a part of a programmer's daily activities. Repository of MRs contains recorded working hours for each request. Three types of working hours exist in the repository: working hours spent in the company, working hours spent on Internet (activities that require Internet access to clients' information system), working hours spent at client side (in client's company). The total number of working hours can be calculated as a sum of these three types of working hours.

4. Maintenance trends analysis

Two sources of data were used for the trend analysis: company documents containing description of organizational structure of the company, and data extracted from the internal repository by using SQL script. Data about programmers' assignments to software applications, and the list of clients with MSA were extracted from company documents. Internal repository contains data about users' requests and other entities that are necessary to track all activities associated to each request. Data was extracted from tables `UserRequest`, `Worker` (programmers), `SoftwareApplication`, `User` (clients) and `WorkOrder`. In table 1 is presented the monthly distribution of solved (completed) maintenance requests used for the analysis in this paper, for clients

Table 1. Monthly distribution of solved maintenance requests

Month	Clients with MSA	Clients without MSA
5.2010.	43	17
6.2010.	37	29
7.2010.	52	24
8.2010.	57	16
9.2010.	42	18
10.2010.	80	18
11.2010.	94	28
12.2010.	88	41
1.2011.	73	49
2.2011.	85	33
3.2011.	85	44
4.2011.	88	30
5.2011.	57	28
6.2011.	60	35
7.2011.	64	30
8.2011.	83	39
9.2011.	49	30
10.2011.	78	54
11.2011.	70	48
Total	**1285**	**611**

with MSA and clients without MSA. The following trends can be drawn: (1) Clients with MSA submit more requests, which is expected since the costs of their requests usually fit the contracted amount in MSA, and (2) The average number of requests per month is 99.79, which practically means that all clients submit approximately four request per working day.

These trends do not provide enough information on maintenance requests' processing. Trends are too general, and therefore not suitable for more detailed analysis. However, these trends prove the high demand for maintenance services. In addition, these trends show that clients with MSA require more maintenance services than clients without MSA. In order to get deeper insight into maintenance trends it is necessary to include details about particular software applications that are maintained, about clients, and about types of maintenance tasks. This analysis enables detection of trends in demands for maintenance by various clients, discovery of distribution of requests per applications, and detection of trends for types of maintenance tasks.

4.1. Monthly trends for maintenance requests per client

Previous analysis shows that clients with MSA submit two times more requests. Therefore, it would be beneficial to find out the distribution of requests per clients, to find out clients with the highest demand for maintenance and based on that to suggest improved versions of MSA that will be tailored to each client or a group of clients. Detailed monthly trends with the number of requests for clients that submit more than five requests per month in average is presented in figure 1. Names of clients' companies are coded with letters *A*,*B*,*C*,*D* and *E* in order to preserve their anonymity according to guidelines for ethical issues in empirical studies in software engineering (Singer &

Table 2. Total and average number of MRs for clients with MSA

	Client A	Client B	Client C	Client D	Client E
Total number	171	170	95	184	142
Average	9.00	8.95	5.00	10.82	9.47

Vinson, 2002). It should be noted that clients D and E have zero requests in the beginning of observed interval because client D started to use software applications in July 2010 and client E in September 2010.

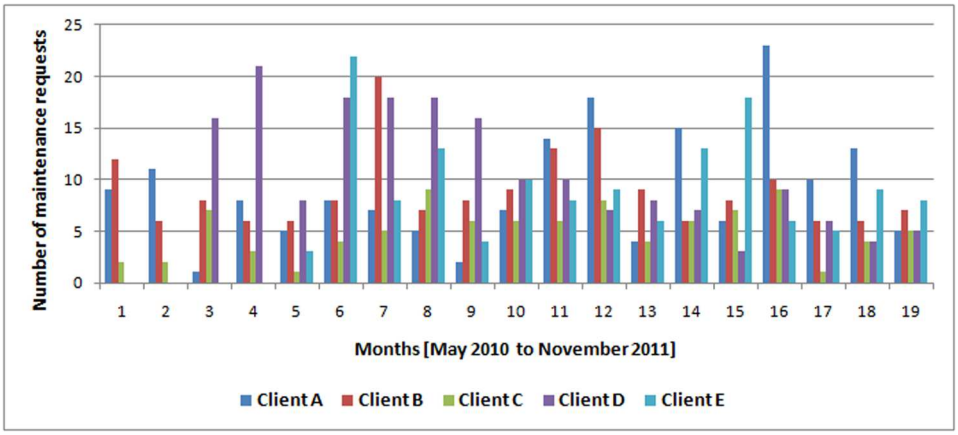

Figure 1. Monthly distribution of MRs for clients that submit more than five requests per month in average

Data presented in figure 1 are related to requests submitted by clients with MSA. Total number of requests, and the average number of requests per month for clients *A*,*B*,*C*,*D* and *E* are shown in table 2. Other clients with MSA submit smaller number of requests, but they submit them in each month.

Clients without MSA submit smaller number of requests than clients with MSA. In addition, they do not submit requests regularly. This means that there are longer periods of time without requests from them. Typical trends for requests submitted by clients (*K*,*M* and *N*) without MSA are presented in figure 2.

Trend analysis of the number of MRs for particular clients can be used for the proactive management of software maintenance activities. In addition, these data can be used also for improvement of policies in MSAs. Since trends for clients with MSA are regular, they could be also used as parameters for estimating future maintenance activities and workload. For clients without MSA it is very hard to draw any conclusion because of irregularity in trends.

Analysis of the number of working hours spent for each client shows the real state of the maintenance workload per client. Figure 3 shows the monthly distribution of working hours for selected clients with MSA.

The average number of working hours for clients *A*,*B*,*C*,*D* and *E* that have MSA per month, and the average number of working hours for clients *K*,*M* and *N* that do not have MSA are shown

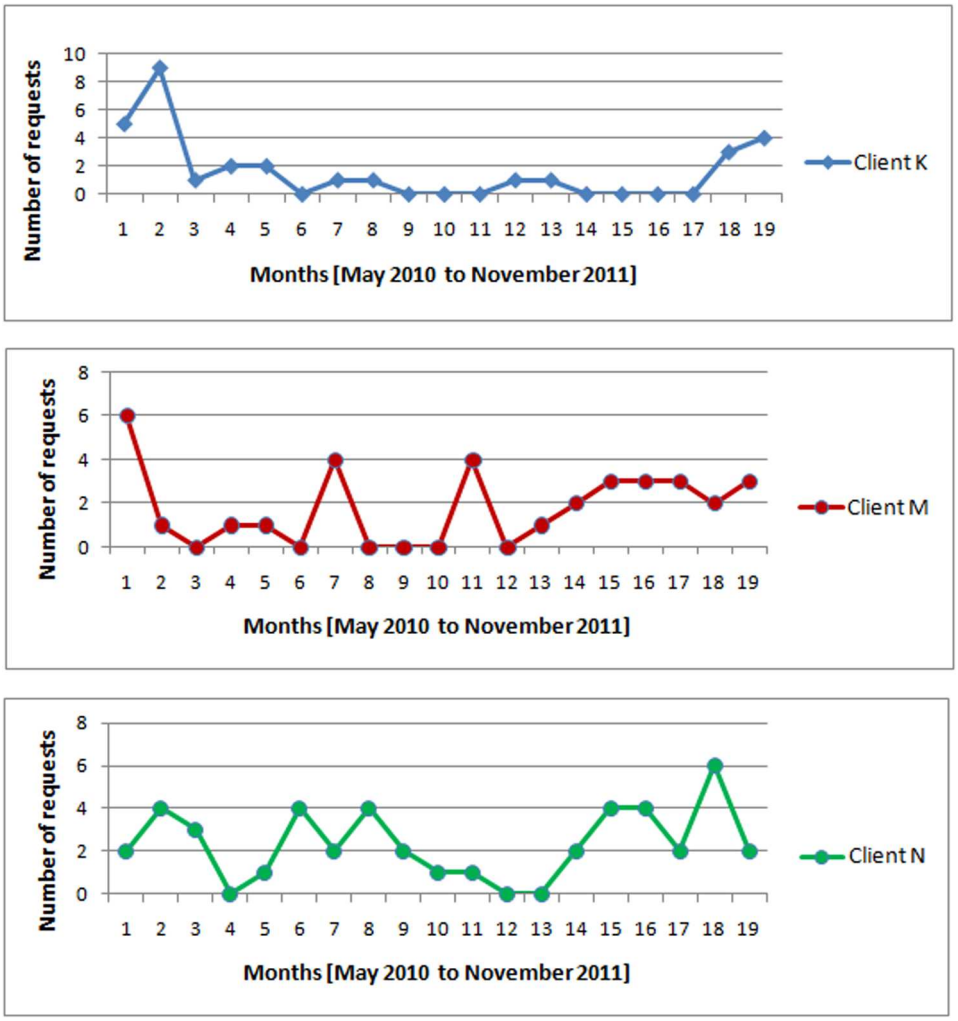

Figure 2. Typical MRs monthly trends for clients without MSA

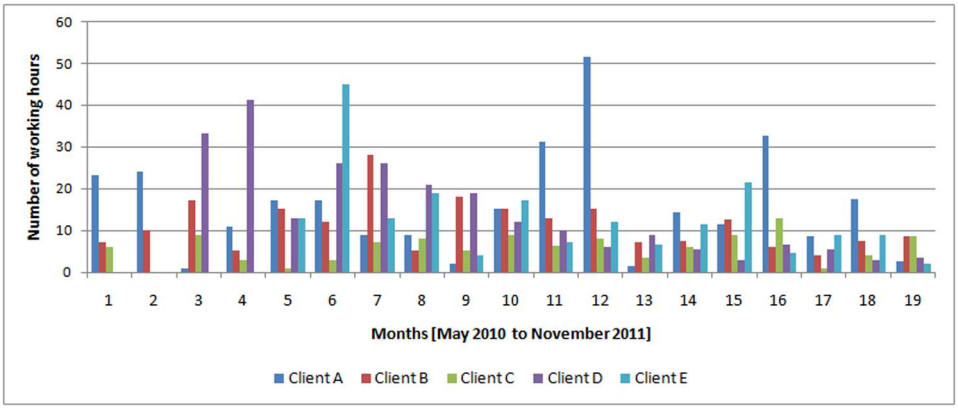

Figure 3. Monthly distribution of working hours for clients with MSA that submit more than five requests per month in average

Table 3. Average number of of working hours per month and per request for clients with MSA that submit more than five requests per month in average

	Client A	Client B	Client C	Client D	Client E
Average number of working hours per month	15.72	11.21	5.81	14.29	12.93
Average number of working hours per request	1.65	1.30	1.16	1¿,17	1.42

Table 4. Average number of of working hours per month and per request for clients without MSA

	Client K	Client M	Client N
Average number of working hours per month	1.13	2.61	1.76
Average number of working hours per request	0.44	0.92	0.64

in tables 3 and 4 respectively.

The first insight into the data related to clients and their requests suggests that clients with MSA consume more time and resources than clients without MSA. This is somehow expected, but directs the thinking towards tailoring appropriate service agreements for particular clients that have not signed agreements yet.

4.2. Monthly trends for maintenance requests per software application

Application portfolio consists of over 30 software applications used by local clients. Organization of maintenance activities is based on assignments of programmers to software applications, which is documented in the company. This means that when somebody receives a request, he/she knows who are potential programmers that should solve it. It is very important to know the distribution of MRs and working hours per software applications in order to improve the maintenance practice.

For that purpose was conducted trend analysis that shows the distribution of maintenance requests per applications, and the distribution of working hours per applications. Analysis disclosed that 75.84 percent of all requests are related to five software applications (named *app1*, *app2*, *app3*, *app4* and *app5*), while 87.39 percent of all requests are distributed to totally nine software applications (see figure 4).

Table 5 shows the average number of working hours per month for five most frequently used software applications, and the average number of working hours per request for these five applications.

4.3. Trends for types of maintenance tasks associated to requests

Classification of maintenance tasks, or maintenance types in the practice is subjected of several studies. From the first typology of software maintenance proposed by Swanson (1976), many authors have proposed different typologies. General agreement among the researchers and practitioners is that maintenance types are: corrective, perfective, adaptive and preventive. However, in practice, software organizations define their typologies according to their needs (Stojanov, 2012*a*).

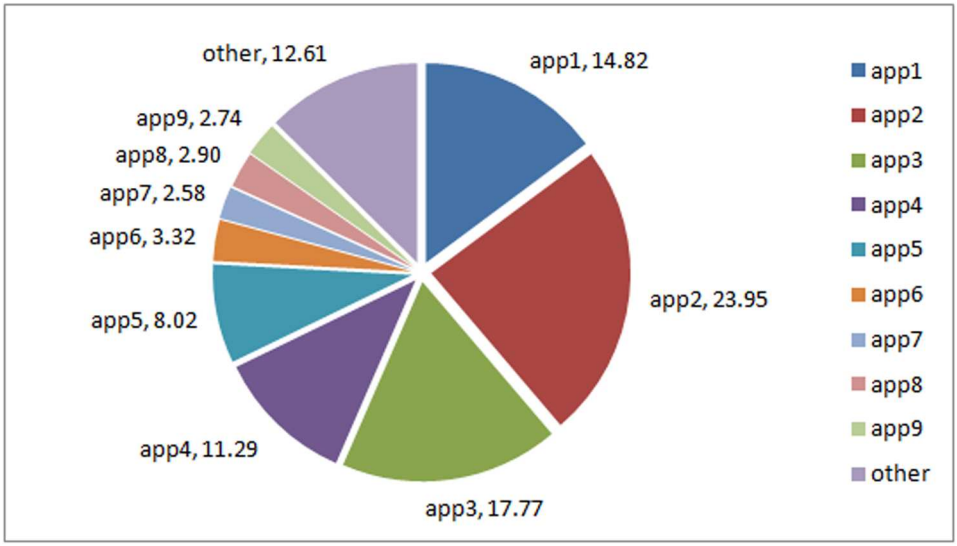

Figure 4. Monthly distribution of MRs per sotware applications

Table 5. Average number of working hours per month and per request for five most frequently used applications

	app1	**app2**	**app3**	**app4**	**app5**
Average number of working hours per month	19.05	33.07	20.91	19.99	9.51
Average number of working hours per request	1.20	1.41	1.18	1.78	1.10

Jones (2010) proposed the list of 23 types of maintenance tasks based on the best practice in industry.

In the selected small company, all maintenance tasks have been recorded in the internal repository. Despite of the large experience in industry, leading experts in the company have not proposed any classification of maintenance tasks based on proposals in available literature, but rather on their own experience and needs. In the repository are defined the following types of tasks: change (any type of change on software products), training, mandatory change (changes proposed by regulative and law), and all other tasks (updates, adaptations). However, more helpful analysis requires more detailed classification of maintenance tasks. For that purpose, general change tasks were manually classified by the company leading programmer in two groups: corrections tasks related to fixing detected faults, and enhancements tasks related to adding new features and other changes not related to faults. Classification was based on detailed description of tasks provided by clients and programmers.

The new classification schema for maintenance tasks is: corrections, enhancements, training, mandatory changes and other. Figure 5 presents trends for types of maintenance tasks in the company. The most of maintenance work is related to enhancing software product capabilities (60.18%), while corrections are related to 23.32% of all maintenance works. All other maintenance tasks contribute with about 10% .

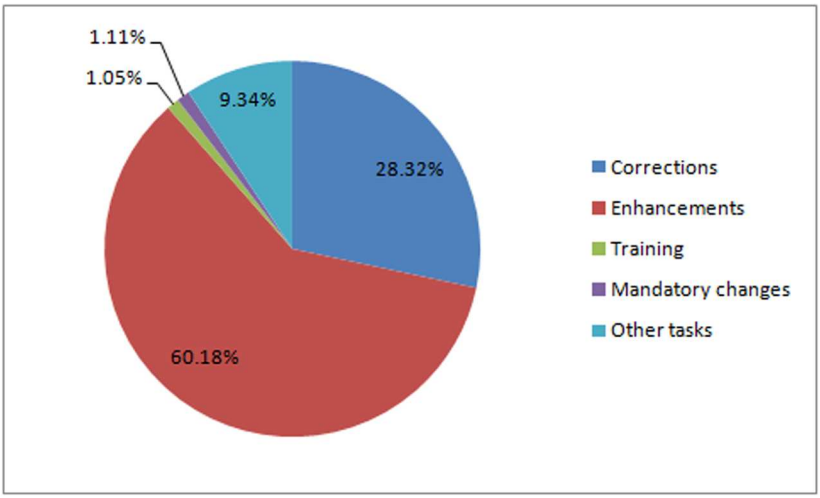

Figure 5. Trends for maintenance tasks

5. Discussion of results in the company

According to assessment plan, feedback meetings (sessions) were regularly organized in the company. Feedback meeting is an effective tool that helps in discussing the state of the assessment process, current findings, and further steps (Dyba *et al.*, 2004; Oktaba & Piattini, 2008). Hattie & Timperley (2007) argued that feedback provides directions for the current practice assessment, learning based on the experience, and further performance improvement.

Company manager and leading programmers participated in the feedback sessions. All sessions were prepared in advance in order to reflect the current state, discussions were type-recorded,

and records were later transcribed and analyzed. Session lasted between 30 and 60 minutes. Practically, sessions were semi-structured, which means that a session plan and initial discussion had been prepared in advance, but discussions during the sessions included many issues that had not been planed.

Discussions related to the analysis of the average number of working hours per clients revealed the following trends:

- Programmers spend approximately six times more time on average for the realization of requests submitted by clients with MSA. According to MSA, clients pay for contracted number of working hours on the monthly basis, and that makes them to feel more comfortable in submitting new requests.
- On the other hand, clients without MSA submits less requests. In addition, their requests require less time in average on the monthly basis.
- The next issues that is obvious is that MRs submitted by clients with MSA consume more time comparing to requests originated from clients without MSA. In addition, clients without MSA are less interested in the improvement of the software applications they use.

Discussions related to the analysis of the average number of working hours per software applications revealed the following trends:

- Software applications labeled with *app1* to *app5* are usually installed as comprehensive business solution for accounting and management of resources in organizations. This explains their dominance regarding the number of requests and consumed working hours. Other applications are not so frequently used and usually are sold as independent software solutions. This implies that the package of these applications should be considered as a candidate for tailoring a special type of MSA for clients that regularly use them, and to offer this option also to other clients.
- For software applications that consume less working hours, solutions that will increase their usability to clients should be identified, which will lead to increase of associated maintenance activities and, therefore, to increased profit to the company. There are few possible directions for further activities that will help in increasing the profit from software applications that are not regularly used: include them in integrated business software solutions, or retire some of them and introduce substitutions that are more attractive to clients.

Discussions related to the analysis of trends in maintenance tasks revealed that the most of the work is related to enhancements and corrections. However, data available in the repository are not suitable for detailed analysis of trends because maintenance tasks have not been properly differentiated.

5.1. Improvement directions

Software process assessment is usually considered in literature as the initial phase of a process improvement (Gray & Smith, 1998). Assessment leads to the identification of key process (practice) elements that need improvements, or towards identification of the strengths and weaknesses that should be considered during improvements' planning (von Wangenheim *et al.*, 2006). Based on the presented trend analysis, the following improvement directions are identified:

- Development of effort estimation models that will be useful in planning programmers' workload. These models will consider monthly distributions of maintenance requests per software applications and per clients and distribution of responsibilities in the company.
- Improvement of planning activities in the company related to distribution of workloads among programmers in order to achieve more efficient and faster processing of maintenance requests.
- Improvements of service agreements for clients. This direction includes proposing different types of MSA that will include various types of software applications. This will lead to portfolio of MSAs that are tailored for special clients' needs.
- Improvements of software applications portfolio management that will consider software applications that are irregularly used, and have very small number of maintenance requests.This direction includes planning the retirement of unsuccessful software and introduction of appropriate substitutions.
- Development of more detailed typology of maintenance tasks that will enable derivation of trends that will cross data about software applications, clients, programmers workload, and maintenance tasks.

6. Limitations and threats to validity

Discussion about internal end external validity, and any other possible limitation is mandatory for empirical studies (Kitchenham *et al.*, 2002).

Internal validity relates to the design of the research, consistency of analysis, and the influence of unexpected sources of bias. Analysis is based on trend analysis techniques that are easy to implement, but requires deeper understanding of the context and full engagement of both researchers (assessors) and company employees that perform the process. This is accomplished by joint work on proposing general improvement and assessment goals, selection of appropriate techniques and methods, and joint analysis of all findings during feedback meetings in the company. The problem with the bias is not addressed since the general goal of the study is practice assessment and improvement and we assume that company employees will provide the full assistance in order to achieve the best results for them. In addition, using rigorous data analysis methods based on traceable data minimized researchers' bias in the research.

The threat to the external validity primarily is related to applicability of this approach in other industrial settings. The approach assumes deeper understanding of the context and involvement

of all company employees in all phases of the research from planning, through collecting and analysing data, to discussing and presenting research findings. Since very small software companies have the similar problems in their business, the approach could be adapted to other small software companies, by considering specificity of their internal organization. The analysis process presented in this study could be also adapted to other, preferably small software companies or small teams. Subsequent applications of this approach would provide evidence about its validity and usefulness.

7. Conclusions

In this paper is presented an approach to software maintenance assessment based on trend analysis and feedback sessions. The study was conducted in a very small software company in Serbia, which is oriented towards local clients. Trend analysis includes analysis of maintenance requests' processing trends with the focus on the number of requests and working hours spent per clients and software applications, as well as simple analysis of maintenence tasks' trends.

The observations and conclusions from trend analysis will be used as directions for process improvement activities in the company. Both technical and organizational issues in the company are subject of improvement based on the results of trend analysis. For example, improvement of client service agreements is the obvious directions for practice improvement related to organizational issues. Improvement of the practice can be achieved also by proposing effort estimation models based on the current trends, such as the model presented by Stojanov *et al.* (2013). The next direction for practice improvement is related to development of more detailed typology of maintenance tasks that will enable analysis of trends for various types of tasks regarding software applications and clients.

The main contribution of the presented approach is related to implementation of assessment method based on trend analysis tailored to a very small software company. The method is based on collecting field data from company maintenance repository, analyzing data by using trend analysis, and identification of relevant conclusions and directions for further improvements of the maintenance practice. The next contribution is detailed presentation of trend analysis as a part of assessment method tailored to specific context, which will be helpful for other small software companies.

The approach is designed for small software companies or teams, and can be tailored to other similar settings. Findings of this research contain lessons that can be used by software practitioners in small software companies in order to assess and improve their decision-making and maintenance requests' processing. On the other hand, researchers could find some useful guidelines how to conduct *light* maintenance assessment based on trend analysis by considering given context.

Further work includes developing a formal model of light assessment approach for software maintenance in very small software companies, and adaptation and implementation of the approach in other similar settings. This will provide the opportunity to replicate the research in order to validate usability of presented approach. The next promising research direction is related to adapting this assessment approach to other processes in small software companies, or to companies that are mostly oriented towards outsourcing of products and services.

Acknowledgement

Ministry of Education and Science, Republic of Serbia, supports this research under the project "*The development of software tools for business process analysis and improvement*", project number TR32044, 2011-2014.

References

Abran, Alain, Bourque, Pierre, Dupuis, Robert, Moore, James W. and Tripp, Leonard L., Eds. (2004). *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. 2004 version ed.. IEEE Press. Piscataway, NJ, USA.

Ahmed, Mohamed F. and Swapna S. Gokhale (2009). Linux bugs: Life cycle, resolution and architectural analysis. *Information and Software Technology* **51**(11), 1618–1627.

April, Alain (2010). Studying supply and demand of software maintenance and evolution services. In: *Proceedings of the 2010 Seventh International Conference on the Quality of Information and Communications Technology*. QUATIC '10. pp. 352–357.

Buglear, John (2001). *Stats means business: a guide to business statistics*. Buttworth-Heinemann. Oxford, UK.

Chatfield, Christopher (1996). *The analysis of time series: an introduction*. Texts in statistical science. fifth edition ed.. Chapman & Hall. London, UK.

Commission, European (2005). *The new SME definition: user guide and model declaration*. Enterprise and industry publications. Office for Official Publications of the European Communities. url: http://ec.europa.eu/enterprise/policies/sme/files/sme_definition/sme_user_guide_en.pdf [accessed 17.2.2013.].

Dyba, Tore, Torgeir Dingsoyr and Nils Brede Moe (2004). *Process Improvement in Practice - A Handbook for IT Companies*. Vol. 9 of *International Series in Software Engineering*. Kluwer Academic Publishers. Norwell, MA, USA.

Fayad, Mohamed E. and Mauri Laitinen (1997). Process assessment considered wasteful. *Communications of the ACM* **40**(11), 125–128.

Fayad, Mohamed E., Mauri Laitinen and Robert P. Ward (2000). Thinking objectively: software engineering in the small. *Communications of the ACM* **43**(3), 115–118.

Gray, E. M. and W. L. Smith (1998). On the limitations of software process assessment and the recognition of a required re-orientation for global process improvement. *Software Quality Control* **7**(1), 21–34.

Hattie, John and Helen Timperley (2007). The power of feedback. *Review of Educational Research* **77**(1), 81–112.

Jones, Capers (2010). *Software Engineering Best Practices*. McGraw-Hill, Inc.. New York, NY, USA.

Junio, Gladston Aparecido, Marcelo Nassau Malta, Humberto de Almeida Mossri, Humberto T. Marques-Neto and Marco Tulio Valente (2011). On the benefits of planning and grouping software maintenance requests. In: *15th European Conference on Software Maintenance and Reengineering*. pp. 55–64.

Kajko-Mattsson, Mira, Ulf Westblom, Stefan Forssander, Gunnar Andersson, Mats Medin, Sari Ebarasi, Tord Fahlgren, Sven-Erik Johansson, Stefan Törnquist and Margareta Holmgren (2001). Taxonomy of problem management activities. In: *Proceedings of the Fifth European Conference on Software Maintenance and Reengineering (CSMR '01)*. Lisbon, Portugal. pp. 1–10.

Kanoun, Karama and Jean-Claude Laprie (1996). Trend analysis. In: *Handbook of Software Reliability Engineering* (Michael R. Lyu, Ed.). Chap. 10, pp. 401–437. IEEE Computer Society Press and McGraw-Hill. New York, USA.

Kenmei, Benedicte, Giuliano Antoniol and Massimiliano di Penta (2008). Trend analysis and issue prediction in large-scale open source systems. In: *Proceedings of the 2008 12th European Conference on Software Maintenance and Reengineering*. CSMR '08. Athens, Greece. pp. 73–82.

Kitchenham, Barbara A., Shari Lawrence Pfleeger, Lesley M. Pickard, Peter W. Jones, David C. Hoaglin, Khaled El

Emam and Jarrett Rosenberg (2002). Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on Software Engineering* **28**(8), 721–734.

Laporte, Claude Y., Alain Renault, Simon Alexandre and Tanin Uthayanaka (2006). The application of iso/iec jtc 1/sc7 software engineering standards in very small enterprises. *ISO Focus* (September), 36–38.

Laporte, Claude Y., Simon Alexandre and Rory V. OConnor (2008). A software engineering lifecycle standard for very small enterprises. In: *Software Process Improvement* (Rory V. OConnor, Nathan Baddoo, Kari Smolander and Richard Messnarz, Eds.). Vol. 16 of *Communications in Computer and Information Science*. pp. 129–141. Springer Berlin Heidelberg.

Lientz, B. P., E. B. Swanson and G. E. Tompkins (1978). Characteristics of application software maintenance. *Communications of the ACM* **21**(6), 466–471.

Mc Caffery, Fergal, Philip S. Taylor and Gerry Coleman (2007). Adept: A unified assessment method for small software companies. *IEEE Software* **24**(1), 24–31.

Oktaba, Hanna and Piattini, Mario, Eds. (2008). *Software Process Improvement for Small and Medium Enterprises: Techniques and Case Studies*. 1st ed.. Information Science Reference - Imprint of: IGI Publishing. Hershey, PA, USA.

Richardson, Ita and Christiane Gresse von Wangenheim (2007). Guest editors' introduction: Why are small software organizations different?. *IEEE Software* **24**(1), 18–22.

Singer, Janice and Norman G. Vinson (2002). Ethical issues in empirical studies of software engineering. *IEEE Transactions on Software Engineering* **28**(12), 1171–1180.

Sousa, Maria Joao (1998). A survey on the software maintenance process. In: *Proceedings of the International Conference on Software Maintenance*. ICSM '98. Bethesda, MD, USA. pp. 265–274.

Stojanov, Zeljko (2011). Discovering automation level of software change request process from qualitative empirical data. In: *Proceedings of the 6th IEEE International Symposium on Applied Computational Intelligence and Informatics, SACI 2011*. Timisoara, Romania. pp. 51–56.

Stojanov, Zeljko (2012*a*). *Software change management methods improvement: Integration of service for specifying change requests in software product model*. Lambert Academic Publishing. Saarbrcken, Germany.

Stojanov, Zeljko (2012*b*). Using qualitative research to explore automation level of software change request process: A study on very small software companies. *Scientific Bulletin of The Politehnica University of Timisoara, Transactions on Automatic Control and Computer Science* **57 (71)**(1), 31–40.

Stojanov, Zeljko, Dalibor Dobrilovic and Vesna Jevtic (2011). Identifying properties of software change request process: Qualitative investigation in very small software companies. In: *Proceedings of the 9th IEEE International Symposium on Intelligent Systems and Informatics, SiSY 2011*. Subotica, Serbia. pp. 47–52.

Stojanov, Zeljko, Dalibor Dobrilovic, Jelena Stojanov and Vesna Jevtic (2013). Context dependent maintenance effort estimation: Case study in a small software company. In: *IEEE 8th International Symposium on Applied Computational Intelligence and Informatics (SACI 2013)*. Timisoara, Romania. pp. 461–466.

Swanson, E. Burton (1976). The dimensions of maintenance. In: *Proceedings of the 2nd international conference on Software engineering (ICSE '76)*. pp. 492–497.

Vasilev, Julian (2012). Guidelines for improvement information processes in commerce by implementing the link between a web application and cash registers. *Theory and Applications of Mathematics & Computer Science* **2**(2), 55–66.

von Wangenheim, Christiane Gresse, Alessandra Anacleto and Clenio F. Salviano (2006). Helping small companies assess software processes. *IEEE Software* **23**(1), 91–98.

Zhu, Tianmei, Yijian Wu, Xin Peng, Zhenchang Xing and Wenyun Zhao (2011). Monitoring software quality evolution by analyzing deviation trends of modularity views. In: *Proceedings of the 2011 18th Working Conference on Reverse Engineering*. WCRE '11. Limerick, Ireland. pp. 229–238.