Trends in software maintenance tasks distribution among programmers: A study in a micro software company

Zeljko Stojanov*, Jelena Stojanov*, Dalibor Dobrilovic*, Nikola Petrov*
* University of Novi Sad/Technical faculty "Mihajlo Pupin", Zrenjanin, Serbia
zeljko.stojanov@uns.ac.rs, jelena.stojanov@uns.ac.rs, dalibor.dobrilovic@uns.ac.rs, nikola.petrov@tfzr.rs;

Abstract— **Software maintenance has been recognized by academicians and practitioners from industry as the most challenging and expensive part in software life cycle. The complexity and high costs of maintenance activities require systematic evidence of all maintenance activities and accurate models for planning and managing them. A common way for analyzing practice in software engineering is based on trend analysis of historical data related to activities and tasks implemented in the past. This paper presents a case study conducted in a micro software company aimed at introducing a schema for classifying maintenance tasks, and identifying trends in software maintenance tasks distribution among the programmers in the company. The discussion of results includes benefits for the company, limitations of the research and implications for academicians and practitioners in industry. The paper concludes with a few promising further research directions.**

I. INTRODUCTION

Software maintenance is a phase in software life cycle that spans all activities after software delivery aimed at sustaining software operable. Although a number of studies reported that software maintenance consumes majority of costs in software life cycle [1,2,3,4,5], most of software companies do not have defined maintenance processes [6]. Software maintenance is typically considered as a set of short term tasks aimed at fixing critical problems or supporting users, but in some cases long term tasks related to major enhancements on existing software occur [1]. Nonetheless, software maintenance tasks are usually considered as boring and programmers try to avoid them [7].

Investments in analyzing trends in maintenance practice and in planning activities can save much time, costs and other resources in software organizations. Since software maintenance is an ongoing process, poor maintenance management and planning may result with several new failures and unpredicted costs for software organizations and their clients [8]. According to April [9], prerequisites for efficient trend analysis of software maintenance are: (1) defined software maintenance processes in terms of timing and goals, (2) tracking of change (maintenance) requests from its sources to implementation on daily basis, (3) precisely defined data collecting procedure for extracting relevant data, and (4) data validating procedure for ensuing that the right data are extracted regarding the proposed measurement objectives.

The first work on defining maintenance processes was conducted by Swanson forty years ago [10]. This early work resulted with the proposed categories of software maintenance: adaptive, corrective, perfective and preventive. The consensus on these categories was documented in IEEE Standard for Software Maintenance (IEEE Std. 1219-1998) [11] and in Guide to the Software Engineering Body of Knowledge (SWEBOK) Version 3.0, published in 2014 [5]. This classification was refreshed by Chapin [12], aimed at reflecting the state of the practice in the field. Chapin introduced several aspects of software maintenance in the classification schema based on the evidence from the reported researches on industrial practice. Kitchenham et al. [13] proposed *a software maintenance ontology* with identified types of maintenance activities. Ruiz et al. [2] proposed a semi-formal ontology with *activity subontology* containing concepts related to classification of maintenance activities, where the basic is modification activity that differentiates to corrections and improvements. Despite the large number of proposed schemas for classifying maintenance, the selection of the most appropriate one depends on the real context.

Based on the previous observations it is evident that analyzing different aspects of software maintenance tasks is real challenge for software organizations. Due to well-known constraints of small software companies related to constrained resources [14,15] and specific business models and objectives [16], investigation of software maintenance tasks is challenging endeavor that can contribute to practice improvements.

This paper presents a study aimed at analyzing trends in software maintenance tasks distribution in a micro software company. The objective of the presented study is to identify trends related to distribution of different types of software maintenance tasks among programmers.

II. RELATED WORK

This section outlines some recent studies on trend analysis in software maintenance. Trend analysis is technical analysis aimed at identifying movements or change patterns of observed objects based on past data. In this context, trend is defined as "long-term change in the mean level", while the real meaning of "long-term" depends on the context [17]. Trend analysis is widely accepted for observing practice and prediction in software engineering since it is easy to comprehend and implement by both engineers and researchers. Common ways of presenting trends in empirical data is by using tables and diagrams with changes of selected variables over time

978-1-5386-3855-2/17/$31.00 ©2017 IEEE

[18]. However, the vast majority of studies reporting trend analysis relate to software development.

Kenmei et al. [19] reported case studies on three large-scale open source software projects over five years: Eclipse, Mozilla and JBoss. The authors proposed a trend analysis approach of change requests based on time series analysis. The empirical data were extracted from version control and bug tracking systems. The study results suggest that time series can be used for predicting new change requests trends.

Trend analysis related to supply and demand of software maintenance services is presented in [9]. The investigated trends relate to distribution of the requests per months, maintenance personnel effort per months, and the distribution of requests and work effort for particular software applications. The trend analysis was conducted as a part of software process improvement project in a software company specialized for developing ERP systems.

The source [20] presents trend analysis of maintenance request (MR) processing in a very small software company. Trend analysis was separated for clients that have Service Level Agreement (SLA) and clients that use maintenance services on demand. Monthly trends of MRs were investigated for the clients that submit majority of MRs and for software applications that were mostly affected by the MRs. A typology of maintenance tasks was not properly developed since several tasks were very hard to classify, which was improved in this study.

III. Case Study

The study was organized and implemented as a joint work of the researchers from academy and the company staff, as a part of a software process improvement project conducted in the selected software company [20]. The company has 3 senior and 3 junior programmers, and according to the classification provided by the European Commission can be classified as a micro company [21]. The company develops and maintains business software for clients in Serbia. The majority of software products have been developed, and operated by the clients, while enhancements and new developments occur when there is a need to meet new user requirements.

A. Context

Trend analysis is based on the data extracted from the internal company repository for tracking all tasks implemented by the programmers. The data were extracted by using SQL scripts and imported in Microsoft Excel for further processing and data analysis. Trend analysis requires data collected for a long period in a systematic manner, and strict following of proposed processes by all stakeholders (clients and programmers). For that purpose totally 2293 tasks were extracted for the period of 19 months starting from February 2013. The detailed analysis of all tasks revealed that 2036 tasks relate to software maintenance, which is 88.79% of all tasks. This indicates the high importance of analyzing maintenance tasks in the company. Monthly trends of maintenance tasks are presented in Fig 1. The most general trend does not reveal any important conclusion, except that the number of tasks slightly increases over time. Compared to the study presented in [20], the number of tasks (each request has one associated task) is greater

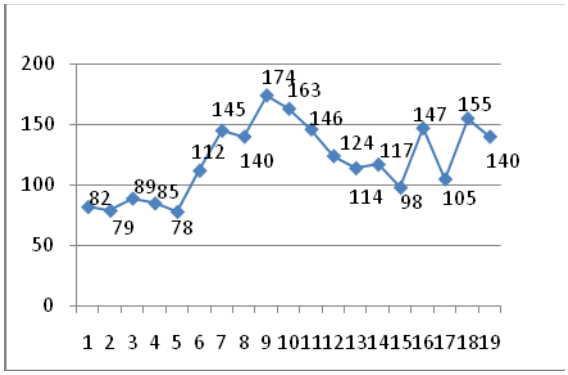

Figure 1. Maintenance tasks monthly trends

by about 150 for a time period of the same duration. The reasons are the increasing number of clients and aging of software products.

A typical maintenance process in the company is defined as a sequence of the following stages: (1) submission of a maintenance request (MR) by a user, (2) scheduling of the MR, (3) acceptance of the MR for implementation by a programmer, (4) implementation of necessary activities to solve the MR, and (5) completing all activities and closing the task. In some urgent cases, MRs are processed immediately, without following all stages in this process.

B. Classification of maintenance tasks

The initial classification of the tasks recorded in the company repository does not provide adequate insight into the real nature of the tasks. For solving this problem, several schemas for classifying software maintenance activities [5,6,12] and maintenance ontologies [13,2] were explored. The adopted schema for classifying maintenance tasks that best fits the state of the practice in the company is presented in Tab. I. The schema was agreed between the company employees and the researchers during several meetings in the company.

TABLE I
TYPES OF SOFTWARE MAINTENANCE TASKS

Type	Abbreviation	Description
Adaptation	A	Adaptation to new hardware platform, operating system or technology
Correction	C	Correction of identified bugs and problems
Enhancement	E	Implementation that changes behavior or feeling during software use
Preventive	P	Changes aimed at preventing possible problems in use
Support	S	Supporting users to efficiently use software

After adopting this schema, all tasks were reexamined and classified by the leading programmer (the most experienced programmer with 27 years of working experience in software industry) and the first author of this paper. Other programmers were consulted in the case of ambiguity. The classification is based on careful reading of the tasks' descriptions imported in the Excel file.

During the classification of the tasks the main dilemma (problem) related to the descriptions that include several different activities. For example, a typical sequence of activities includes: modifying software, installing software, testing software, and providing training. The question is: which activity to choose as dominant for classifying the task? The primary decision was to select the maintenance activity that was judged as the most important based on the programmers' opinions, and to classify the task according to that activity.

C. *Maintenance tasks distribution*

The analysis started with identifying to which class of maintenance best fits each recorded task. This analysis resulted with the distribution of maintenance tasks per identified categories, as it is presented in Tab. II.

TABLE II
DISTRIBUTION OF SOFTWARE MAINTENANCE TASKS ACCORDING TO THE PROPOSED CLASSIFICATION SCHEMA

Type	Number	Share [%]
Adaptation	22	1.08
Correction	289	24.02
Enhancement	1050	51.57
Preventive	8	0.39
Support	467	22.94

It is evident from the data presented in Tab. II that the majority of tasks relate to activities aimed at ensuring the continuous and reliable use of software products in the clients' business environments. More than half of all tasks relate to enhancing software functionalities to satisfy business needs of clients. A typical enhancement task can be described as:

> *Add a report by customers for a selected period and goods.*

Corrective maintenance accounts for about one-quarter of all tasks. Supportive maintenance tasks, including trainings and consultations, accounts also for about one-quarter of tasks. These two categories of tasks relate to solving problems reported by users, implemented by fixing software if necessary or by providing relevant consultancy expertise.

Adaptive and preventive maintenance occur rarely, which is the sign of mature software products that fit to business needs of the clients. Preventive maintenance tasks are mostly implemented for aligning software to changed regulative at the national level or standards in the relevant branch of economy.

The next important trend for maintenance tasks relates to the tasks distribution among programmers, as it is presented in Tab. III. For that purpose senior programmers are marked as S1, S2 and S3, while junior programmers are marked as J1, J2 and J3. Trend analysis based on Tab. III shows that senior programmers solve totally 783 tasks (38.46%), while juniors solve 1253 tasks (61.54%). Further, juniors are mostly engaged at corrective and enhancive tasks (approximately two-thirds of tasks), while seniors solve the vast majority of adaptive tasks (72.73%) that require more comprehensive knowledge of hardware, software and their integration in the clients business

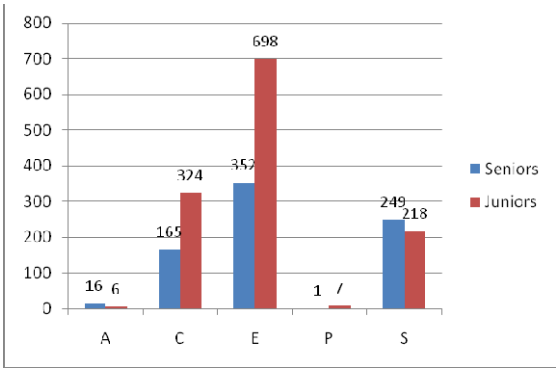

Figure 2. Maintenance tasks distribution on senior and junior programmers

TABLE III
DISTRIBUTION OF SOFTWARE MAINTENANCE TASKS AMONG PROGRAMMERS

	Maintenance tasks categories					
Programmer	A	C	E	P	S	Total
S1	6	82	177	1	111	377
S2	4	59	129	0	92	284
S3	6	24	46	0	46	122
J1	0	88	151	4	49	292
J2	6	235	541	3	167	952
J3	0	1	6	0	2	9

environments. The distribution of maintenance tasks on senior and junior programmers for the proposed classification schema, summarized in Tab. III, is presented in Fig. 2.

IV. DISCUSSIONS

A. *Benefits for the company*

The first benefit for the company is proposed and improved schema for classifying maintenance tasks, which enables accurate classification of all tasks. The tasks were formerly classified based on the experience of senior programmers in the company [20], without consulting relevant proposals in literature [12,6,5] or standards [11]. Improvement of classification schema relates to introduction of supportive, adaptive and preventive tasks, which reflects the real nature of the maintenance tasks.

The most important class of the tasks that was introduced relates to support activities that formerly included only training activities, but now includes all consultative tasks that help users to solve problems with software products and data. Supportive tasks now include 22.94% of all maintenance tasks, comparing to previous classification where trainings cover only 1.05% of all tasks. In this schema, all tasks are classified in five introduced classes, comparing to 9.34% of non classified tasks identified in [20].

Insight into tasks' distribution among the programmers can be further used for improving scheduling of maintenance tasks and planning further activities. Improvement directions relate to development of models

for predicting efforts per task types and workload of the programmers by including spent working hours, or models for controlling evolution of software products such as tracking major or minor modifications, and planning withdrawals from use (software retirement).

The lessons learned during analyzing maintenance trends can be used for analyzing trends in other business segments in the company, such as software development, software testing, human resources management or organizational issues.

And finally, on the basis of the observed flaws in the recording of data (quality, accuracy and consistency), the company may improve its evidence about business activities by relying on systematic and accurate historical data [22], which is necessary for reliable decision making based on identified trends.

B. *Limitations of the study*

Limitations and quality of the study and results are commonly discussed in terms of internal and external validity [23].

Internal validity was ensured through selection and careful implementation of trend analysis techniques by considering the real working context in the company. Joint work of the company staff and researchers on selecting the most appropriate data sets and data analysis techniques minimizes bias and ensured the relevance of the results for the company.

External validity assumes transferability of study results in other settings, which is quite questionable since the study was designed for the selected company. However, identification of the most suitable classification schema and adequate trend analysis techniques can be used in other small software organizations since majority of them share the same problems in everyday practice.

C. *Research implications*

The results of this study contain guidelines for classifying maintenance tasks that can be used by software practitioners in industry for assessing their maintenance practice and improving their decision making and planning in software maintenance.

Researchers can also find guidelines for organizing lightweight analysis of software maintenance practice by using trend analysis and considering specific working context, or may implement similar techniques for analyzing other segments of practice.

V. CONCLUSIONS

This study presents the research aimed at improving a classification schema for software maintenance tasks and distribution of maintenance tasks among programmers in a local micro software company. The approach, tailored for the selected company, is based on joint work of the company staff and the researchers, and appropriate statistical methods for analyzing trends in data sets extracted from the company local repository of tasks.

The main contribution of the study is the improved classification schema for software maintenance tasks, which is aligned to the real practice in the company and relevant literature and standards. Based on the new classification schema, tasks are reexamined and classified, which enables identification of trends for each programmer or groups of them. The next contribution relates to detailed presentation of trend analysis, which can be adapted to other studies (projects) aimed at investigating the state of the practice, preferably in small software organizations.

Several research directions can be distinguished. The first assumes validation of the new classification schema on new maintenance data sets in the company. Similarly, the approach can be used for analyzing other tasks or processes in software life cycle (e.g. evolution of software requirements specification, or development and testing tasks). The next research direction relates to developing a formal and adaptable model for classifying maintenance tasks based on recognition of relevant keywords in the tasks' descriptions, which can be used for developing a software tool for automatic or semi-automatic maintenance tasks classification. Tailoring and implementing this approach in other small software companies is also promising research direction, which will help in judging usability of the presented trend analysis approach.

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, 2011-2017.

REFERENCES

[1] R. D. Banker and S. A. Slaughter, "A field study of scale economies in software maintenance", *Management Science*, vol. 43, no. 12, pp. 1709-1725, 1997. doi: 10.1287/mnsc.43.12.1709.

[2] F. Ruiz, A. Vizcaíno, M. Piattini, and F. Garcia, "An Ontology For The Management Of Software Maintenance Projects", *International Journal of Software Engineering and Knowledge Engineering*, vol. 14, no. 3, pp. 323-349, 2004. doi: 10.1142/S0218194004001646.

[3] G. A. Junio, M. N. Malta, H. de Almeida Mossri, H. T. Marques-Neto, and M. T. Valente, "On the Benefits of Planning and Grouping Software Maintenance Requests", In *Proceedings of the 15th European Conference on Software Maintenance and Reengineering*, 2011, pp. 55-64. doi: 10.1109/CSMR.2011.10.

[4] F. J. Pino, F. Ruiz, F. García, and M. Piattini, "A software maintenance methodology for small organizations: Agile_MANTEMA" *Journal of Software: Evolution and Process*, vol. 24, issue 8, pp. 851–876, 2012. doi: 10.1002/smr.541.

[5] Pierre Bourque and Richard E. (Dick) Fairley (Editors). *Guide to the Software Engineering Body of Knowledge, Version 3.0, SWEBOK*. IEEE. 2014.

[6] A. April and A. Abran, *Software Maintenance Management: Evaluation and Continuous Improvement*. Hoboken, NJ, USA: IEEE Computer Society & Wiley, 2008.

[7] P. Stachour and D. Collier-Brown, "You Don't Know Jack About Software Maintenance", *Communications of the ACM*, vol. 52 no. 11, pp. 54-58, 2009. doi: 10.1145/1592761.1592777.

[8] R. D. Banker, S. M. Datar, C. F. Kemerer, and D. Zweig, "Software Errors and Software Maintenance Management", *Information Technology and Management*, vol. 3, no. 1-2, pp. 25-41, 2002. doi: 10.1023/A:1013156608583.

[9] A April, "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)*, 2010, pp. 352-357. doi: 10.1109/QUATIC.2010.65.

[10] E. B. Swanson, "The dimensions of maintenance", In *Proceedings of the 2nd international conference on Software engineering (ICSE '76)*, 1976 pp. 492-497. San Francisco, CA, USA.

[11] IEEE Computer Society, *IEEE Standard for Software Maintenance, IEEE Std 1219-1998*. New York, NY, USA: IEEE, 1998. doi: 10.1109/IEEESTD.1998.88278.

[12] N. Chapin, "Software maintenance types-a fresh view", In *Proceedings of 2000 International Conference on Software Maintenance*, 2000, pp. 247-252. San Jose, CA, USA. doi: 10.1109/ICSM.2000.883056.

[13] B. A. Kitchenham, G. H. Travassos, A. von Mayrhauser, F. Niessink, N. F. Schneidewind, J. Singer, S. Takada, R. Vehvilainen, and H. Yang, "Towards an ontology of software maintenance", *Journal of Software Maintenance: Research and Practice*, vol. 11, issue 6, pp. 365-389, 1999.

[14] G. Coleman and R. V. O'Connor, "An investigation into software development process formation in software start-ups, *Journal of Enterprise Information Management*, vol. 21, issue 6, pp. 633-648, 2008. doi: 10.1108/17410390810911221.

[15] Basri, S. and O'Connor, R., "Understanding the Perception of Very Small Software Companies towards the Adoption of Process Standards", in Riel et al (Eds), *Systems, Software and Services Process Improvement, CCIS*, Vol. 99, Springer-Verlag, pp. 153-164, 2010.

[16] I. Richardson, C. G. von Wangenheim, "Guest Editors' Introduction: Why are Small Software Organizations Different?", *IEEE Software*, vol. 24, no. 1, pp. 18-22, 2007.

[17] C. Chatfield, *The analysis of time series: an introduction, 5th edition*. Texts in statistical science. London, UK: Chapman & Hall, 1996.

[18] G. Robertson, R. Fernandez, D. Fisher, B. Lee, and J. Stasko, "Effectiveness of Animation in Trend Visualization", *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1325-1332, 2008. doi: 10.1109/TVCG.2008.125.

[19] B. Kenmei, G. Antoniol, and M. di Penta, "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)*, 2008, pp. 73–82. Athens, Greece.

[20] Z. Stojanov, D. Dobrilovic, and J. Stojanov, "Analyzing Trends for Maintenance Request Process Assessment: Empirical Investigation in a Very Small Software Company", *Theory and Applications of Mathematics & Computer Science*, vol. 3, no. 2, pp. 59-74, 2013.

[21] European Commission. *The new SME definition: User guide and model declaration*. Luxembourg: Office for Official Publications of the European Communities. 2005.

[22] T. Girba and S. Ducasse, "Modeling history to analyze software evolution, *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 18, issue 3, pp. 207-236, 2006. doi: 10.1002/smr.v18:3.

[23] D. I. Sjoberg, T. Dyba, and M. Jorgensen, "The Future of Empirical Methods in Software Engineering Research", In *Proceedings of 2007 Future of Software Engineering (FOSE '07)*, 2007, pp. 358-378. Minneapolis, Minnesota, USA. doi: 10.1109/FOSE.2007.30.