Discovering Software Process Deviations Using Visualizations

Anna-Liisa Mattila[1(✉)], Kari Systä[1], Outi Sievi-Korte[1], Marko Leppänen[1], and Tommi Mikkonen[2]

[1] Tampere University of Technology, Tampere, Finland
{anna-liisa.mattila,kari.systa,outi.sievi-korte,marko.leppanen}@tut.fi
[2] University of Helsinki, Helsinki, Finland
tommi.mikkonen@helsinki.fi

Abstract. Modern software development is supported by a rich set of tools that accumulate data from the software process automatically. That data can be used for understanding and improving software processes without any manual data collection. In this paper we introduce an industrial case where data visualization of issue management system was used to investigate software projects. The results of the study show that visualization of issue management system data can really reveal deviations between planned process and executed process.

Keywords: Software visualization · Mining software repositories

1 Introduction

Various business information systems are focal for corporate management, and often companies utilize metrics as critical success indicators for their business [1]. So, in management of any process, both access to valid process data and the ability to understand the meaning of the data are essential.

Building automated data collection frameworks requires time and effort and is an investment for the company [2], but collecting data manually from the employees is a tedious and error prone effort. Fortunately in software development the effort required to access the data can be reduced significantly as many tools, such as version control and issue management systems, already automatically collect some data [3]. Thus, utilization of this ready-at-hand data could make process analysis more feasible for software companies.

Raw data items or numbers can rarely illuminate the analyst. Therefore, visualization methods are used to get a better overall picture of the organization and its business. When visualizations are available, various stakeholders can enjoy improved transparency to the actual status and react to possible issues faster. The usefulness of these visualizations is not limited to managers only, as everybody can benefit from good visualizations of the progress and properties of the project. This follows the spirit of Andon boards that are used to notify

H. Baumeister et al. (Eds.): XP 2017, LNBIP 283, pp. 259–266, 2017.
DOI: 10.1007/978-3-319-57633-6_18

management, maintenance, and other workers of a quality or process problems in the Toyota Production System [4].

Many kinds of visualizations are used to show different aspects of software engineering process. Standard visualisation methods in project planning, such as *Gantt charts* [5] and Scrum *burndown charts* [6] can be used as well as workflow visualizations such as *Kanban board* [7,8]. The current state of the project can be communicated to the developer team using *radiators* and *dashboards* [8,9]. When in software process management and improvement, it is important to know what happened in the past and for this purpose various timeline based visualizations have been developed [10,11]. The idea in these methods is to show what happened in the past based on data. This kind of visualizations can be used as a tool in retrospective meetings [10], and during the development to spot abnormalities in the process [11].

In this paper we present experiences on visualizing data from software repositories. We explore the software process in two industrial projects. The paramount goal of our work is to help stakeholders, especially project managers, to observe the execution of software process to find deviations from the planned process, and to detect possible problems in the projects.

2 Research Process

The main research questions of the study are: (1) *Can we show deviations from the assumed software process by visualizing data gathered from software repositories?* (2) *Is the visualization of project data helpful for keeping track of the projects?* To answer these questions, we decided to study software projects where we could access the data starting from the beginning of the project. Issue management system was chosen as our data source as it is used for managing and reporting the software projects. The research process is presented in detail in Fig. 1.

Selected Cases. The cases studied are two industrial projects of a Finland-based multinational large-sized company involved in software R&D. We selected the company based on their interest towards the research. The company representatives selected the studied projects with the following constraints: suitable

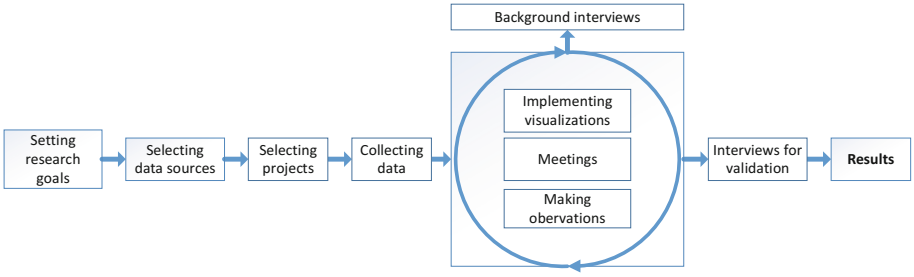

Fig. 1. The research process.

data are available from the beginning of the project, the project is currently in the development phase, and selected projects are comparable with each other.

Both of the studied projects are sub-projects of a larger software entity. In this paper, we refer to the projects as project A and project B. In project A, a software platform is developed whereas in project B a user interface for the software is developed. Both projects have 5–10 team members; the team composition varies based on the current need. Most of the team members are developers, but in team B there are also dedicated persons for testing and user experience design.

The projects use JIRA[1] for issue tracking. The guidelines for using JIRA are the same in both projects. The projects follow the same software development process, namely Scrumban [12], which is a hybrid of Scrum and Kanban. As the projects have uniform practices and processes, we assume that the project data are comparable between the projects.

The data collection period was from the start of the projects till the beginning of January 2015. The projects had started in 2013 – project A in May and B in August. The data sets we used were anonymized by the company representatives. The data were delivered in text format and contained only the information necessary for visualization and analysis – for example person names, JIRA comments or issue names were not visible to us.

Participants. We had eight participants from the case company. A manager of the larger project entity which the studied projects are part of (P1), a person responsible of the realization of agile ways of working in both projects and who was also a former developer in project B (P2), three developers from project A (P3, P4, P5), and three developers from project B (P6, P7, P8).

Four researchers participated to the research by studying the project data, developing the visualization tool, and participating the meetings with the case company.

The visualization tool. To empirically examine the relationship between project data and the perceived state of the project we built a software visualization tool. We chose to utilize timeline as the visualization format because it enables us to easily explore how projects evolved over time and it is used for similar purposes in other studies as well [11,13]. We held several meetings with participants P1, P2, and P3 from the case company to receive feedback from the visualization. The visualization was developed in an iterative manner where we fine-tuned the visualizations based on the received feedback.

The main element of the visualization is to show lifespans and state changes of issues reported in JIRA. Through the lifespan visualization we can observe which issues have been open for a long time and through which states the issue is finally resolved. Detailed figures of the visualization are provided with other additional material on https://github.com/pervcomp/DSPDUV.

Interviews. We held two interview sessions for developers in the projects studied. The first interviews were held at the beginning of the research process to

[1] JIRA – Project management system, https://www.atlassian.com/software/jira.

gain feedback from the initial version of the visualization and get deeper knowledge of the case projects. In the first interview we had three participants: P2, P3, and P6. The second interviews were held four months later to validate our observations made from the visualizations and to gather feedback from the visualization. In the second interview we had seven participants: all the three people interviewed in the first round (P2, P3, and P6) were interviewed again along with two more developers from both projects (P4, P5, P7, and P8). We selected the themes in a fashion that allowed us to (i) validate the assumptions considering the visual observations, and to (ii) reveal the ways of working in the projects as well as possible problems in the team and project.

The interviewing sessions were conducted as follows. Each interview began by discussing the background of the interviewee and continued to the discussion about ways of working, challenging issues in the team work and the project's current status. We showed the visualization to the interviewee during the last part of the interviewing session and asked the interviewee to observe and interpret the visualization. Finally, we discussed the observations made by researchers together with the interviewee to identify potential misinterpretations and to determine causes of the observed issues. Interviewees were also asked to give feedback from the visualization and tell if they thought the visualization is a useful tool for managing projects. The duration of the interviews varied from 30 to 60 min. All interviews were recorded and written notes were made. The interviews were conducted by one researcher.

3 Results

The results are based on studying the visualizations of project data and interviews. The data visualized from the projects were *bug reports*, *epics*, and *stories*. We made assumptions of status and ways of working from the visualizations. The table of assumptions made is available on https://github.com/pervcomp/DSPDUV with the visualizations and other additional material.

Bugs. When comparing the views that show the lifespans and resolution rate of bug reports we noticed that the resolution rate of bug reports was higher in project A than in project B. When interviewing the participants, we found out that in project A bug fixes were prioritized over implementing new features. Prioritizing the bug fixes over new features was not an actual policy of the software process but an agreement within the team thus in project B similar convention was not applied.

The long life spans and increasing amounts of bug reports in project B could be a sign of technical debt or bad architectural decisions but also relate to problems in organizing and reporting work. Based on the interviews we learned that in project B there was technical debt as they had built the project directly on top of their initial prototype, which should have been just a throw away prototype. In project A the initial prototype was discarded. There were also problems in organizing and reporting work in project B.

Epics. The projects differed in how they used epics to plan greater entities. In project B only three epics were closed during the data collection period and all open epics were in their initial state. In project A epics were closed and opened in a more regular pattern. Based on this we could assume that in project B the role of epics in planning was not clear and they were not used systematically, which was also proven to be the case based on the interviews. Based on the process and instructions given to the teams they were supposed to use epics similarly when planning work.

Stories. When looking at the projects individually we assumed that both projects had problems in organizing and reporting work. The assumption was made based on long lifespans and increasing amounts of open stories that were visible in the visualizations. Also the long lifespans and high amount of open bug reports in project B supported this assumption for project B. Based on the interviews there were problems in organizing work. In both projects the product owner's role was not clear. In project A the product owner was not committed to organize the backlog, and in project B there was no product owner.

Usefulness of the visualization. To get feedback on use of the visualization for tracking the projects, we asked if the interviewees considered the visualization useful. All of the interviewees agreed that the visualization we presented is practical in tracking the projects as it shows clearly the issues which have been open for a long time. Most of the interviewees mentioned that the visualization would be especially useful for the project managers but also for them selfs.

4 Threats to Validity

Wohlin et al. [14] state four different categories when considering threats to validity - conclusion validity, internal validity, construct validity and external validity. We will deal with those that are particularly relevant to our study.

Threats to conclusion validity are concerned with issues that affect the ability to draw the correct conclusion about relations between the treatment and the outcome of an experiment. The threats most concerning our study have to do with "fishing" for a particular result and reliability of measures. In the start of the research process we did not expect any results, but were simply curious about what could be learned by visualizing software project data. Thus, all the observations made are purely drawn from what could be seen and without prejudice. Furthermore, the visualizations were interpreted together with company representatives, who would correct false assumptions. Additionally, the interviews were designed to reveal possible overlooked information from the visualizations.

Reliability of measures, in turn, involves the measured data (from the repositories) and verbal information (interviews). The data itself is visualized "*as is*", without any human involvement required in between, so it is valid. The interview questions, in turn, were designed in a way that would allow as open answers as possible and for the interviewer to also perform follow-up questions. Naturally, the wording of the questions is still always critical, and for example a pilot study of the interviews could have been beneficial.

Threats to internal validity concern causality and threats to conclusions about relationships between treatment and outcome. In our experiment the most relevant threat regards *selection*, i.e., selecting the subjects, in our case the interviewees from the company, and how volunteering might affect the results. The interviewees were selected so that we had at least one developer and one person in charge of the process for both projects, who answered questions on both interview rounds, thus ensuring a versatile perspective of the project. For the second round the subjects were selected among developers based on who had the time. Thus there was no direct volunteering, which might affect results, and also selection was not made on any other criteria than having different roles, which should ensure a true view of the project. However, there was also no means to control the backgrounds of interviewees either.

Finally, construct validity concerns generalizing the result of the experiment. The most relevant threat to this study are *hypothesis guessing* and *evaluation apprehension*, both having to do with whether the interviews can be trusted. We argue that evaluation apprehension is not a concern, as all interviewees were willing to discuss the problems in their projects, and did not attempt to hide them from the researcher. As for guessing the hypothesis, we did not show the visualization to the interviewees until in the end of the interview, so all answers were purely given based on the questions.

The final category of threats given by [14] relates to external validity are conditions that limit the ability to generalize the results of the experiment to industrial practice. This does not concern us, as our cases were from the industry, and thus we can argue that the results already reflect industrial practice. However, more research need to be done for generalization of results.

5 Discussion and Conclusions

The first research question we addressed in Sect. 2 was *"Can we show deviations from the assumed software process by visualizing data gathered from software repositories?"*. Using the visualization we could note differences in practices between projects that should have had the same practices. We were also able to interpret from the visualization that there were problems in the software process. We learned that problems related to organizing work shows well in the visualization of issue management data. We also learned that different problems show differently. The problems in planning and reporting are visible in long lifespans of issues as well as different kinds of patterns in creating issues where as technical debt may be visible in bug report lifespans and creation rate.

Our second research question was: *"Is the visualization of project data helpful for keeping track of the projects?"*. Based on the feedback we can conclude that the visualization is a useful tool for project managers. Furthermore, we noticed that the visualization raised questions and interest in participants to discuss about the state of the projects. The visualization creates a good common ground for such discussion as it shows empirical evidence.

We have developed the visualization tool further based on the feedback received from the case company. We have also done first experiments using the

visualizations in teaching software engineering. As a future work we will investigate the use of the tool for other industrial projects as well as for open source projects to validate our findings and evaluate the tool further.

References

1. Menzies, T., Zimmermann, T.: Software analytics: so what? IEEE Softw. **30**(4), 31–37 (2013)
2. Robbes, R., Vidal, R., Bastarrica, M.: Are software analytics efforts worthwhile for small companies? The case of Amisoft. IEEE Softw. **30**(5), 46–53 (2013)
3. Mäkinen, S., Leppänen, M., Kilamo, T., Mattila, A.L., Laukkanen, E., Pagels, M., Männistö, T.: Improving the delivery cycle: a multiple-case study of the toolchains in Finnish software intensive enterprises. Inf. Softw. Technol. **80**, 175–194 (2016)
4. Liker, J.K.: The Toyota Way. Esensi (2004)
5. Gantt, H.: Work, Wages and Profit. The Engineering Magazine (1910)
6. Schwaber, K., Beedle, M.: Agile Software Development with Scrum, 1st edn. Prentice Hall PTR, Upper Saddle River, NJ (2001)
7. Kerzazi, N., Robillard, P.N.: Kanbanize the Release Engineering Process. In: 2013 1st International Workshop on Release Engineering (RELENG), pp. 9–12. IEEE (2013)
8. Paredes, J., Anslow, C., Maurer, F.: Information visualization for agile software development. In: 2014 Second IEEE Working Conference on Software Visualization (VISSOFT), pp. 157–166. IEEE (2014)
9. Baysal, O., Holmes, R., Godfrey, M.W.: Developer dashboards: the need for qualitative analytics. IEEE Softw. **30**(4), 46–52 (2013)
10. Bjarnason, E., Hess, A., Doerr, J., Regnell, B.: Variations on the evidence-based timeline retrospective method: a comparison of two cases. In: 2013 39th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), pp. 37–44. IEEE (2013)
11. Lehtonen, T., Eloranta, V.P., Leppänen, M., Isohanni, E.: Visualizations as a basis for agile software process improvement. In: 2013 20th Asia-Pacific Software Engineering Conference (APSEC), vol. 1, pp. 495–502. IEEE (2013)
12. Ladas, C.: Scrumban - Essays on Kanban Systems for Lean Software Development. Modus Cooperandi Press, USA (2009)
13. Bjarnason, E., Svensson, R.B., Regnell, B.: Evidence-based timelines for project retrospectives - a method for assessing requirements engineering in context. In: 2012 IEEE Second International Workshop on Empirical Requirements Engineering (EmpiRE), pp. 17–24. IEEE (2012)
14. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering: An Introduction. Kluwer Academic Publishers, Norwell (2000)