

Software Process Evaluation from User Perceptions and Log Data

Damjan Vavpotič¹  | Saimir Bala²  | Jan Mendling²  | Tomaž Hovelja¹ 

¹Faculty of Computer and Information Science, University of Ljubljana, Ljubljana, Slovenia

²Institut für Informatik, Humboldt-Universität zu Berlin, Berlin, Germany

Correspondence

Saimir Bala, Institut für Informatik, Humboldt-Universität zu Berlin, Rudower Chaussee 25, 12489 Berlin, Germany.
Email: saimir.bala@hu-berlin.de

Funding information

Einstein Foundation Berlin, Grant/Award Number: EPP-2019-524

Abstract

Companies often claim to follow specific software development methodologies (SDM) when performing their software development process. These methodologies are often supported by dedicated tools that keep track of work activities carried out by developers. The purpose of this paper is to provide a novel approach that integrates analytical insights from both the perceptions of SDM stakeholders and software development tools logs to provide SDM improvement recommendations. This paper develops a new process improvement approach that combines two significantly different sources of data on the same phenomenon. First, it uses a questionnaire to gather software development stakeholder SDM perceptions (managers and developers). Second, it leverages process mining to analyze software development tools logs to obtain additional information on software development activities. Finally, it develops recommendations based on concurrent analysis of both sources. Our novel process improvement approach is evaluated in three directions: Does the presented approach (RQ1) enable managers to gain additional insights into employees' performance, (RQ2) deliver additional insights into project performance, and (RQ3) enable development of additional SDM improvement recommendations? We find that integrated analysis of software development perception data and software development tools logs opens new possibilities to more precisely identify and improve specific SDM elements. The evaluation of our novel process improvement approach follows a single case study design. Our approach can only be used in enterprises in which software development tools logs are available. The study should be repeated in different cultural settings. We practically show how concurrently analyzing data about developer SDM perceptions and event log data from software development tools enables management to gain additional insights in the software development process regarding the performance of individual developers. The main theoretical contribution of our paper is a novel process improvement approach that effectively integrates data from management and developer perspectives and software development tools logs.

This is an open access article under the terms of the Creative Commons Attribution-NonCommercial-NoDerivs License, which permits use and distribution in any medium, provided the original work is properly cited, the use is non-commercial and no modifications or adaptations are made.

© 2022 The Authors. Journal of Software: Evolution and Process published by John Wiley & Sons Ltd.

KEYWORDS

case study, evaluation, software development methodology, software development tool logs, stakeholder perceptions

1 | INTRODUCTION

Software development methodologies (SDMs) highly influence the management of software projects. Adopting a specific methodology can determine to what extent performance goals such as cost, quality, and timeliness are achieved.¹ Existing studies show²⁻⁴ that the adoption of SDMs is critical for success of development teams, but not easy to apply in practice. More specifically, it is often the case that a specific methodology is either applied partially or incorrectly, is not suitable for the enterprise due to incompatible technical characteristics, or has unsatisfactory impact on software quality, cost, and development time.^{5,6} Therefore, understanding the performance of specific SDM elements is crucial for management's capability to continuously improve software development processes.⁷ These SDM elements include SDM activities as well as artifacts produced and tools used in these activities.

In order to gain knowledge of the current state of SDMs within software companies, newest approaches in literature have focused on the analyses of their constituting SDM elements.⁸⁻¹⁰ These studies focus on perceptions of various stakeholders about software process activities, tools, and roles and neglect software development process data from supporting tools. This additional source of information regarding development process became available in recent years with widespread use of tools used in software development activities like issue tracking, requirements management, and test management. Such tools store valuable data about actual execution of the development process and can provide additional insights into SDM and its elements, thus complementing stakeholder perceptions.¹¹⁻¹⁴ What these recent works do not address is to what extent both sources of information complement each other.

In this paper, we propose a novel approach to evaluate a process improvement approach of the performance of SDM activities through concurrent analysis of stakeholder perceptions and relevant software development tools logs. In this way, we can consider both sources of insights about performance of specific SDM activities and consequentially improve organizational learning.¹⁵ Based on this analysis, we develop SDM improvement recommendations. We evaluated our approach through a case study in an Austrian SME software development company.

This work seeks answers to the following research questions:

- RQ1. Which new insights into employees' performance for management does the proposed approach provide?
- RQ2. Which new insights into project performance for management does the proposed approach provide?
- RQ3. Which new insights does the concurrent analysis of stakeholder SDM perception data and software development tools logs provide and which SDM improvement recommendations would be otherwise unavailable?

The rest of the paper is organized as follows. Section 2 provides background on the state of the art. Section 3 describes the methodology. Section 4 applies the methodology on a case study from industry and shows the results. Section 5 discusses the results and their implications. Section 6 concludes the paper.

2 | STATE OF THE ART

This paper builds upon existing research on SDM evaluation and process mining. In the following, we briefly introduce these fields of research.

2.1 | SDM evaluation

Existing research shows that using SDMs improves productivity of software development enterprises and quality of the developed software.¹⁶⁻¹⁸ This is achieved by increasing enterprises' ability to transfer knowledge between employees, systematically manage software development process, and so forth.^{9,19-21} However, it is not enough that an enterprise only describes its SDM in a document; the developers need to use it consistently in their everyday work. The use of SDM was often a topic of research in the last decades, because SDM adoption among software developers was relatively low and the developers often preferred different ad hoc approaches.²²⁻²⁴ According to Destefanis et al.,²⁵ developers are the key factors for the success of a software development process, not merely as executors of tasks, but as protagonists and core of the whole development process. Therefore, understanding their perceptions is of key importance in SDM improvement.

The use of SDMs in enterprises can be analyzed with the help of different approaches based on technology acceptance model, theory of planned behavior, and so forth.^{26–28} One of the foundational theories in the field of SDM evaluation is diffusion of innovations,²⁹ according to which SDM is considered as innovation that developers adopt.^{30–32} To obtain information about the studied SDM and its elements, the aforementioned studies focused on SDM perceptions held by different stakeholders, namely, developers, managers, and customers to measure characteristics like level of use, assimilation, social and technical suitability, developer satisfaction, and impact on performance.^{7,10,29,33–35} By considering opinions of multiple stakeholders, our approach fits well with agile SDM that emphasize team autonomy, decentralized decision making, flexible scope, and managing a high degree of requirement changes.^{36,37}

Detailed insight into SDM and its elements can be gained by comparing the perceptions of different stakeholders regarding the same SDM and/or its element.⁹ Another important theoretical development in the field was that SDM should be studied at the level of its constituent elements like activities, tools, roles, and produced documents and not only as a whole. This improved the understanding of suitability of studied SDM elements for a certain development team and enabled comparison between the studied SDM elements; thus, allowing enterprises to better pinpoint problematic elements of SDM, prepare focused improvements, and examine the link between a specific SDM element and overall project success.^{9,33} Such development is in line with findings in the field of situational method engineering^{8,38–40} that recommends the construction of a custom SDM from those SDM elements that fit with characteristics of certain development team and other situational factors (internal and external enterprise's environment). One of the key goals of these improvements was to tackle low SDM adoption rates that process and practice improvement initiatives often suffer from.⁴¹ Such low adoption rates clearly indicate the inadequacy of existing SDM evaluation models and the need to develop better ones, for instance based on the analysis of log data.

2.2 | Analysis of event logs

Process mining is a discipline that has gained great popularity in the last decade. The goal of process mining is to provide fact-based insights and support process improvement.⁴² On a broader context, process mining can be considered as the missing link between traditional model-based process analysis and data-driven techniques such as data mining and machine learning. Van der Aalst,⁴³ defines four partially overlapping perspectives of a business process that are widely used in business process modeling (BPM) community. First, the *time perspective* aims at analyzing time and frequency of process events. Second, the *case perspective* aims at identifying properties of process cases. Third, the *organizational perspective* aims at analyzing the event log to gain transparency on the resources involved in the process. Fourth, the *control-flow perspective* aims at analyzing the different variations of the process, that is, in which order its constituting activities are carried out in real life. These aspects can be, for example, related to the control flow (i.e., the various steps used by the company for generating value), resources (i.e., handover of work among the different process participants), activities (i.e., how the work is broken down into several tasks), and data (i.e., which artifacts are produced and consumed by the process).

Process mining is becoming widely adopted. Although researchers can use general purpose data-mining tools like R, Orange, and IBM SPSS, a considerable number of commercial tools such as Celonis,⁴⁴ Disco,⁴⁵ Apromore,⁴⁶ minit,⁴⁷ and LANA Process Mining⁴⁸ have been developed. A comparison of the current process mining tools can be found in Viner et al.⁴⁹ In academia, log analysis is an established endeavor when it comes to gaining insights into software development. Recent works^{50–54} demonstrate the potential for understanding the different perspectives of processes. Process mining has also been considered as a research method⁵⁵ with applications for analyzing process improvement methods⁵⁶ and reporting software experiments.⁵⁷

Continuous evolution of data analyses techniques (such as process mining), their adoption in new domains and their integration into other research methods in different mixed-methods approaches, indicate the future opportunities that novel SDM evaluation approaches can exploit. In this context, the necessity and importance of our research that attempts to integrate current state of the art process mining approaches and SDM evaluation approaches becomes clear, because the integration can deliver insights that neither of the two approaches alone can provide.

3 | PROPOSED APPROACH

3.1 | Overview

In this section, we present our novel process improvement approach. The development of our approach followed principles from design science⁵⁸ and situational method engineering⁵⁹ as research methods. The proposed approach builds on the established SDM evaluation approaches. Gill et al.⁸ emphasize the need for project to meet stakeholders' expectations thus analysis of different stakeholders' perceptions must be an integral part of a modern SDM evaluation approach. Vavpotič and Bajec¹⁰ showed the importance of evaluating SDMs by their constituent parts and not

only as a whole. Vavpotič and Hovelja⁷ presented the usefulness of considering economic success metrics of software development in SDM evaluation. However, unlike existing approaches that base their SDM evaluation solely on stakeholder perceptions, the proposed approach complements stakeholder perceptions with data from software development tools logs. The approach comprises four phases as shown in Figure 1. To our knowledge, this is the first attempt to combine perceptions and log data in SDM evaluation. Next, we elaborate on the phases of the approach.

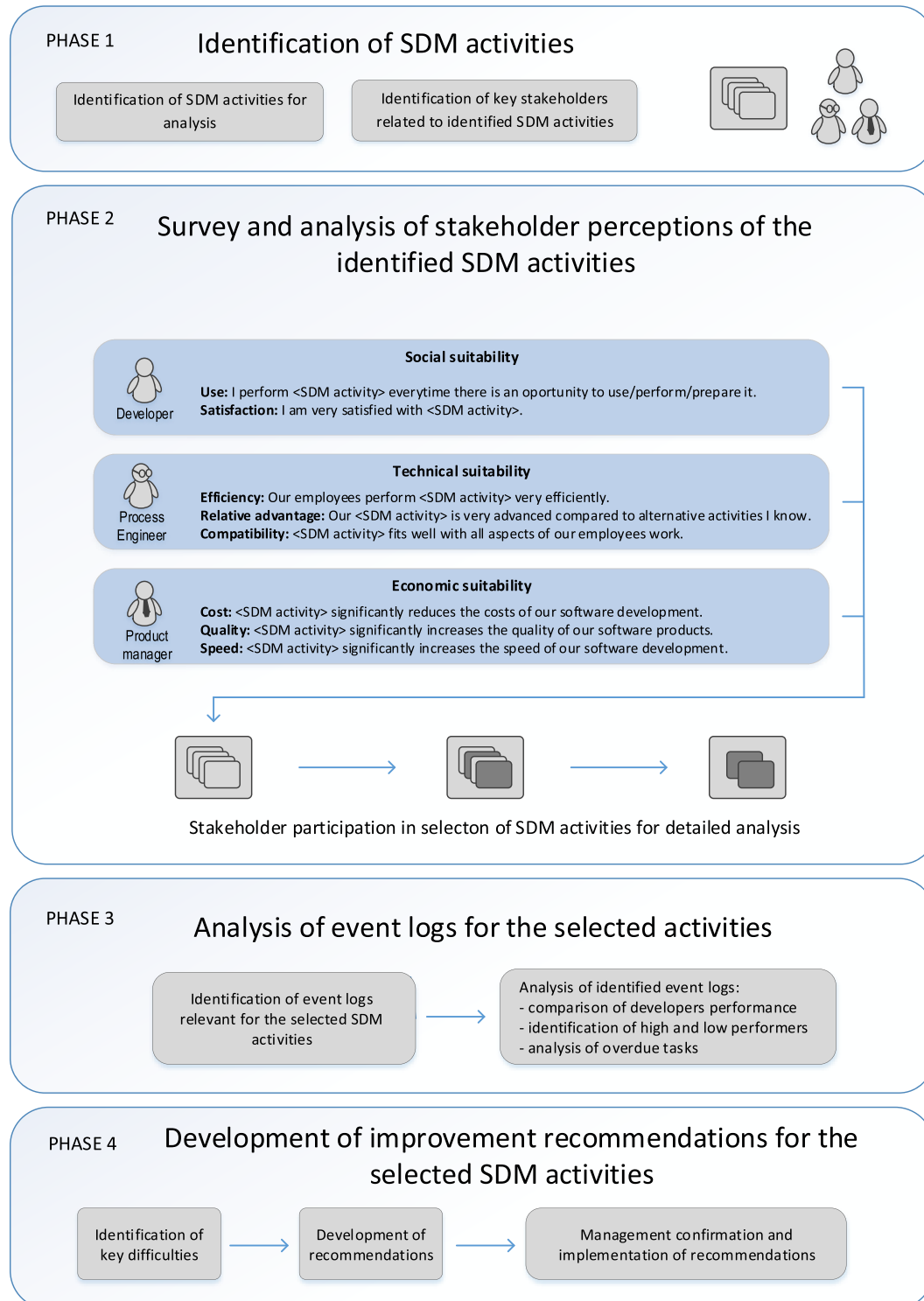


FIGURE 1 The proposed SDM evaluation approach. SDM, software development methodology

3.2 | Phase 1—Identification of SDM activities

In Phase 1, SDM activities for evaluation are first identified and cataloged with the help of key stakeholders knowledgeable about their SDM (e.g., lead developers and managers). In identification phase, we focus on activities because log data are typically organized around them. Roles are considered indirectly as specific activities are typically performed by a specific role. If an enterprise formally follows a specific standard SDM (like Scrum, IBM Rational Unified Process, and Kanban), the cataloging starts with a predefined list of typical SDM activities. However, the key stakeholders still have to point out which SDM activities are actually used and identify possible additional activities in the enterprise and only those are included in the final catalog. After the SDM activities are cataloged the management has to identify the actual stakeholders performing these activities in a specific project that will be evaluated. Managers responsible for a certain project and/or product should be the ones to evaluate the SDM activities used in project and/or product development from managerial perspective. Similar logic is also applied to other project stakeholders.

3.3 | Phase 2—Survey and analysis

Phase 2 starts with the creation of questionnaires that include each SDM activity cataloged in Phase 1 based on the template questions (shown in Figure 1). Different questionnaires are created for each stakeholder. In line with the established measures from studies presented in the literature review, the developers evaluate SDM activity social suitability (average of use and satisfaction). Similarly, product managers evaluate SDM impact on iron triangle measures of economic suitability (average of cost reduction, speed increase, and quality increase). The questions are written in form of statements where answers are given on 7-point Likert scale.

To analyze stakeholder perceptions, we position SDM activities in a multidimensional space where each dimension represents a specific stakeholder perspective. It is typically difficult to further improve SDM activities that are considered highly beneficial by all stakeholders since they are all already satisfied with them. On the contrary, SDM activities that are considered unbeneficial by all stakeholders have high potential for improvement, because it is likely that most stakeholders will support their change. The SDM activities where perceptions of stakeholders are very low and/or greatly differ require further examination where log analysis (Phase 3) has an important role to identify appropriate improvement actions.

3.4 | Phase 3—Analysis of software development tools logs for selected activities

In Phase 3, log analysis is performed focusing on selected SDM activities. Log analysis provides additional information about certain SDM activities such as the time developers spent their execution of this activity and the quality of the execution. To guide the log analysis, we followed the PM² methodology.⁶⁰ PM² consists of six stages, namely (1) planning—in the goals and the target processes are identified, (2) extraction—in which trace data of the identified processes are collected, (3) data processing—in which data are transformed into event logs, (4) mining and analysis—in which the event logs are analyzed by means of process mining techniques, (5) evaluation—in which results of process mining are interpreted and linked to improvements ideas, and (6) process improvement and support—in which the improvement ideas are implemented.

When it comes to analyzing SDMs, PM² can be adapted as follows. First (1), we start the log analysis with identification of software development tools logs relevant for the selected SDM activities. Second (2), event logs are gathered from the respective information systems. Then, an inspection of the information contained in these logs is performed to confirm that relevant information about resources and activities is present. Third (3), data processing is performed. In fact, different software development tools have their own logging mechanisms and often use customized representations of event logs. Such logs are rarely in the format needed by process mining tools. Therefore, preprocessing is required in order to transform these logs into a format that can be further analyzed by process mining techniques.^{61,62} Therefore, the data are stored into a form of a flat table with explicitly labeled notions of case, activity, and timestamp. This representation can be easily transformed into the standard XES⁶³ format for event logs or be directly consumed process mining tools.

Fourth (4), we analyze instances of activities preformed during development that are described by at least three aspects: activity type (bug fixing, implementing new code, etc.), actor (employee ID1, employee ID2, etc.), and timestamp (date1, date2, etc.). These three aspects enable us to analyze identified software development tools logs to compare developers' performance, identify high and low performers and analyze overdue activity instances for specific types. To conduct these analyses depending on log structure, different process mining techniques can be used like analyses of case durations, filtering on users, and variant analyses. Moreover, some software development tools logs contain additional information like priority of specific types of activities, pointers to related or preceding activities, and detailed unstructured descriptions of preformed activities and comments. If needed, these can be used to gain additional insights into the software development process as a whole as well as into specific SDM activities.⁶⁴ As we focus on a single case, we use process mining to support exploring these different aspects related to the selected SDM elements.

Finally, (5 & 6) several analyses artifacts resulting from the applications of the aforementioned process mining techniques such as process maps, case/activity/user frequency reports, and durations of activities are collected. For instance, for the SDM element of bug fixing, we can observe how often this was worked on by a certain user and whether it took the user one or more sprints to complete his task. These kinds of insights are then compared with the perceptions collected via the questionnaire to help resolving conflicting perceptions. For instance, it may be the case that management does not perceive correctly the time and effort spent by developers on bugs. Equipped with evidence-based results from event logs, it is possible to align the perceptions of the different stakeholders.

3.5 | Phase 4—SDM improvement recommendations

In this phase, systematic and comprehensive data from Phases 2 and 3 are presented to external and/or internal SDM experts (if enterprise has sufficient competences). Their task is to identify key difficulties of the current development process based on the concurrent analysis of stakeholders' SDM perceptions and software development tools logs. Next, based on identified key difficulties they prepare recommendations for process improvement using their knowledge and experience. Different problem-solving techniques like brainstorming, simulation, what-if-analysis, and creativity techniques that help to support the act of improving can be used in this phase. An example of recommendations developed by such approach is presented in Section 4. Finally, the recommendations need to be presented, discussed with, and confirmed by the management. After management acceptance of the proposed recommendations, a timeline for their implementation can be developed.

4 | CASE STUDY

4.1 | Case study description and research methodology

To evaluate our novel process improvement approach, we conducted a single case study. The proposed approach was tested in an Austrian SME software development company located in Vienna. The company can be considered a typical central European software development SME using agile SDM. The company develops a software platform in the field document composition, workflow, and document distribution. Their customers come from eight different central and southern European countries. Seven developers and a product manager worked on company's customer communications management (CCM) platform in the studied year. The product manager oversaw the quality of the product features, software process speed, and cost. All developers participating in our study were experienced developers having worked as members of the same team for at least 2 years.

The company uses Scrum,⁶⁵ an agile, lightweight, iterative, and incremental methodology often used in software development. Scrum organizes work in iterations called sprints. The length of sprint is set in advance and is typically between 1 week and 1 month, with 2 weeks being the most common. As in most Scrum implementations each sprint starts with sprint planning and ends with sprint review (presenting work to stakeholders) and sprint retrospective (identifying issues and improvements related to the development process).

The company uses the JIRA issue tracking software to monitor tasks completion by their developers. We obtained data for two activities (implementing new code and bug fixing) from 26 sprints for five developers working on the studied activities between 2016-01-11 and 2017-01-08. The JIRA event log was preprocessed to generate a flat table. In tabular datasets, we created additional attributes to track the number of performed activities. We aggregated the information into 2-week intervals reflecting company sprints. This helped us understand if the studied activities were completed on schedule (in one sprint) or not. Furthermore, JIRA allows us to track activity completion rate on the level of the whole project and on the level of individual developers.

An exploratory single case study design was employed to assess the proposed approach.⁶⁶ Such a research design is appropriate to capture the circumstances and conditions of an everyday or commonplace situation.⁶⁷ The studied company represents a typical SME in the software development industry. To collect the data, we conducted interviews with management, directly observed their workday, surveyed all developers working on the observed project, and collected the corresponding JIRA event logs. The study results were presented and validated by the SME's management.

4.2 | Identification of SDM activities (Phase 1)

In the first phase, we identified 10 key activities (Table 1) of the company's software development process by interviewing the lead developer and product manager. They also specified the other developers who worked on this project. The identified 10 activities cover key parts of the SDM used in the company. These include requirements specification, planning, coding with unit testing, and maintenance. We also identified JIRA event logs that were used in one or more of the identified 10 activities in the observed period.

TABLE 1 Identified activities and availability of JIRA event logs

Development activities	Availability of JIRA event logs related to the activity
Defining list of specifications (in Excel)	No
Defining feature specifications	Yes
Using prepared feature specification	Yes
Using prepared bug specification in Jira	Yes
Estimating (planning poker)	No
Assigning from backlog to sprint	No
Implementing new code	Yes
Bug fixing	Yes
Sprint planning (bi-weekly)	Yes
Stand-ups (daily meetings)	No

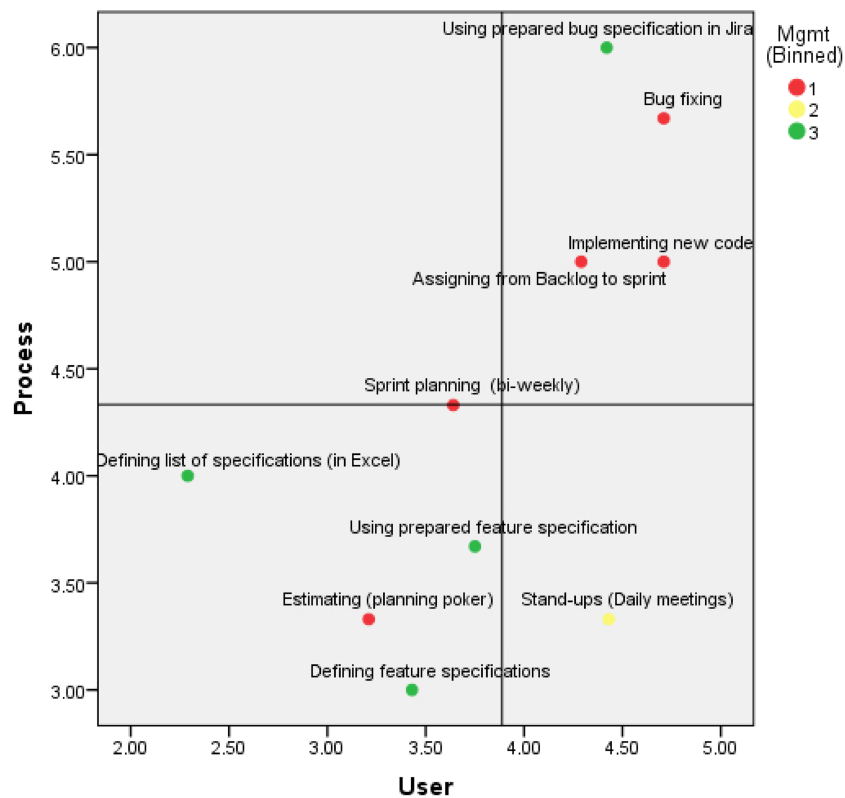


FIGURE 2 Scatter chart of stakeholder views on the identified development activities

4.3 | Survey and analysis of stakeholder perceptions of the identified SDM activities (Phase 2)

Based on the template questions (Figure 1), we created two separate questionnaires, one for the seven developers and one for the product manager. For the purpose of the analysis, we visualize the results of the survey of the two identified stakeholders on a scatter chart comprising the developers' perspective as a horizontal dimension and the product manager's perspective as a vertical dimension (Figure 2).

To better understand the views of the two stakeholders on specific SDM activities, four quadrants were formed in the scatter chart by using the average scores of developers' and management perspectives. The four quadrants follow the logic suggested by Vavpotič and Hovelja⁷: the first quadrant contains managerially and socially unsuitable SDM activities, the second quadrant contains managerially unsuitable but socially suitable SDM activities, the third quadrant contains managerially suitable but socially unsuitable SDM activities, and the fourth quarter contains managerially and socially suitable SDM activities. Each SDM activity is presented by a point in one of the four quadrants of the scatter chart. The red color points represent the five activities that were selected for detailed evaluation. Management limited the selection of activities for analysis and

improvement to maximum half of all SDM activities (i.e., five activities) due to limited capabilities and resources available for the improvement processes. The selection was done as follows. All SDM activities that were unbeneficial for both stakeholders (sprint planning and estimating) were selected for further analysis from the first quadrant (I). From the second quadrant (II), the two SDM activities (implementing new code and bug fixing) that consume significantly more work hours per sprint were selected. Whereas from the third quadrant (III), one SDM activity was selected (defining list of specifications) that was significantly lower graded by the developers.

User stories described in *defining feature specifications* activity were prioritized and assigned to a list (backlog) in *defining list of specification* activity. During the interviews, we learned that the main reason for dissatisfaction of developers with *defining list of specification* activity was that management was often adjusting prioritizations of user stories to ensure customer satisfaction when customer priorities changed. Due to regular shifts in customers' priorities, this approach caused difficulties for developers who often had to de-prioritize user stories related to architecture and other technical aspects. The dissatisfaction of the developers was further exacerbated by the fact that they felt team autonomy and team ability to plan its work was affected. Moreover, the list of specifications was maintained in MS Excel, which caused additional difficulties for the developers due to delayed list synchronization.

The *estimating (planning poker)* and *sprint planning (bi-weekly)* activities were performed by the team each sprint as follows. First, the team performed *estimating (planning poker)* to determine the effort needed for each task. Next, in *sprint planning (bi-weekly)*, these tasks were divided among the team members. According to the management, these activities were among the least efficient activities. The management was dissatisfied by often exceeded time limits as tasks were not completed in a single sprint as planned. The dissatisfaction of the team was mostly the result of increased but unsuccessful management pressure to raise performance.

Implementing new code and *bug fixing* activities were timewise the main activates in the development process. Although the developers were satisfied with their performance, seeing themselves as highly capable performers and valuable employees, the management did not share their view and considered these two activities similarly inefficient as *estimating (planning poker)* and *sprint planning (bi-weekly)*. The main cause of management dissatisfaction was again poor compliance with task completion time limits set by the team.

4.4 | Analysis of JIRA event logs (Phase 3)

We analyzed JIRA event logs related to four out of the six selected development activities with available JIRA event logs to gain new insights especially where the views of management and developers differed. *Defining list of specification* was maintained in MS Excel, thus no JIRA event logs were available. First, we analyzed the frequency of bugs and tasks not completed in each examined sprint (26 sprints = 1 year) to get additional insights in *estimating (planning poker)* and *sprint planning (bi-weekly)* activities. Figure 3 shows the number of not completed bugs per sprint, and Figure 4 shows the number of not completed tasks per sprint. In both cases, there is considerable variability in unfinished tasks/bugs between sprints. This exposes planning and performance difficulties as well as an unstable development process.

To gain additional insights in *bug fixing* and *implementing new code* activities, we compiled Table 2. It shows the individual developer performance breakdown in number and percentage of bugs fixed and new code implementations (tasks completed) in a single sprint and those completed in more than one sprint.

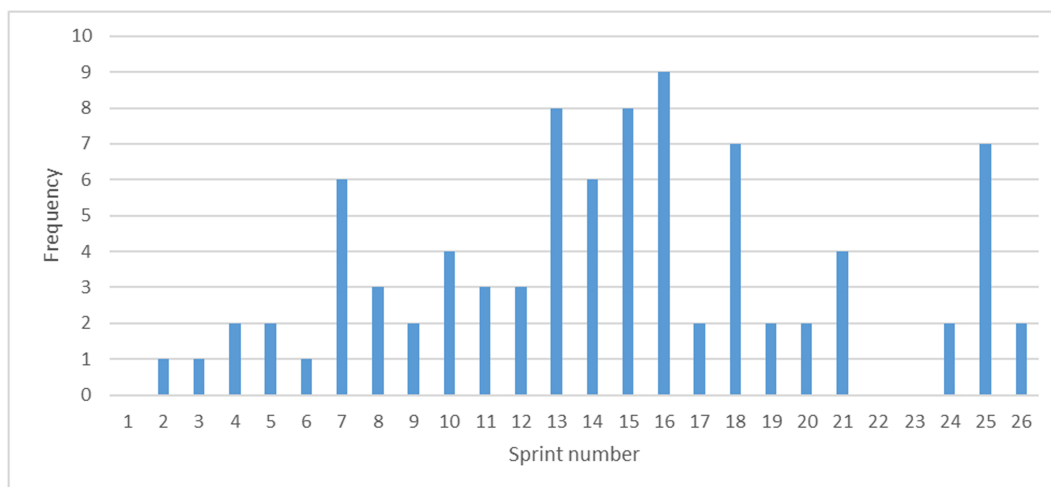


FIGURE 3 Frequency of bugs not completed in a single sprint by sprint number

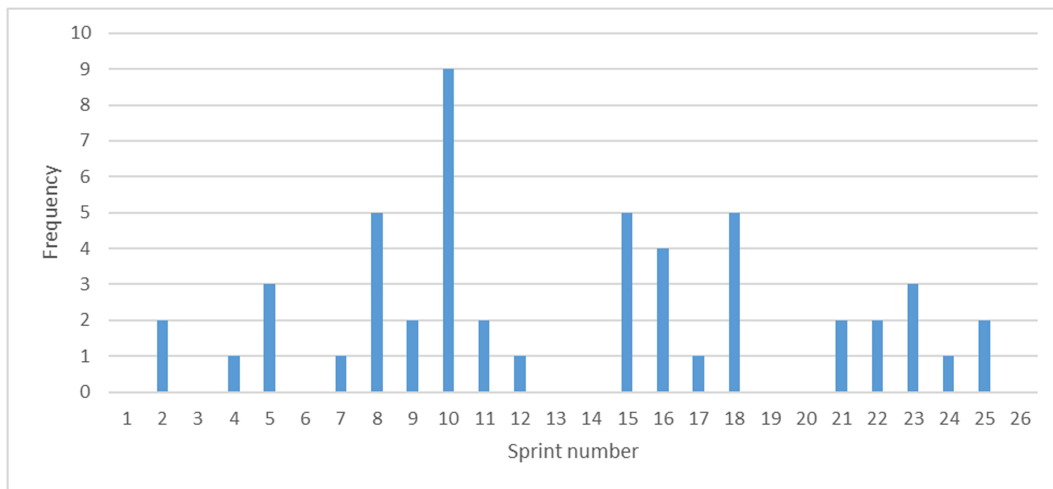


FIGURE 4 Frequency of tasks not completed in a single sprint by sprint number

TABLE 2 Bugs and tasks completed in a single sprint and in more than one sprint for each developer

		Developer 1	Developer 2	Developer 3	Developer 4	Developer 5	Total
Number of bugs	Completed in a single sprint	22	22	17	39	30	130
	Completed in more than one sprint	12	16	17	33	9	87
	Total	34	38	34	72	39	217
Percentage of bugs	Completed in a single sprint	64.7%	57.9%	50.0%	54.2%	76.9%	59.9%
	Completed in more than one sprint	35.3%	42.1%	50.0%	45.8%	23.1%	40.1%
Number of tasks	Completed in a single sprint	^a	78	18	16	19	131
	Completed in more than one sprint	^a	10	15	8	18	51
	Total	^a	88	33	24	37	182
Percentage of tasks	Completed in a single sprint	^a	88.6%	54.5%	66.7%	51.4%	72.0%
	Completed in more than one sprint	^a	11.4%	45.5%	33.3%	48.6%	28.0%

^aDeveloper 1 was performing only *bug fixing* (bugs), whereas the other four developers were performing *implementation of new code* (tasks) and *bug fixing*.

The analysis of performance clearly shows significant differences between individual developers. Additionally, there were significant differences in on-time completion rate between bug fixing and implementation of new code, showing that if a developer is a top performer in one activity he can still struggle in the other. The most extreme cases were Developers 2 and 5. On one hand, Developer 2 was the best performer in *implementation of new code* but was below average in *bug fixing*. On the other hand, Developer 5 was the best performer in bug fixing, but the worst in implementation of new code. Developers 3 and 4 were below average in both activities. Additionally, Developer 3 was relatively worse in *implementing new code*. Developer 1 was above average in *bug fixing*; however, he did not participate in *implementing new code*.

This information derived from log analysis allowed us to gain the following additional insights for the four selected development elements with available JIRA event logs.

The logs show that there is significant variability in task and bugs completion rates between developers; thus, the management perception that developers' performance was not satisfactory was only partially correct. JIRA event logs helped us determine that there are some great performers, some average performers and some poor performers. To tackle this problem successfully, these differences have to be better managed in order to improve *estimating (planning poker)* and *sprint planning (bi-weekly)* activities. If this issue is properly addressed, management can also expect improvements in *implementing new code* and *bug fixing* activities.

4.5 | Development of improvement recommendations for selected SDM activities (Phase 4)

Based on the above presented analysis in Phases 2 (Section 4.3) and 3 (Section 4.4) of the proposed approach, the following improvement recommendations were prepared by external experts for implementation by management (see Table 3).

TABLE 3 Identified key difficulties and SDM improvement recommendations

Identified key difficulties		SDM improvement recommendations	
D1	Adjusting prioritizations of user stories to ensure customer satisfaction when customer priorities changed	R1	Currently, a manager is acting as an intermediary between developers and the customer. Management needs to start to act as connector that encourages and helps the team to establish direct collaborations between them and the customer's product owners (lead users). This would increase the team's autonomy and ability to plan its work and at the same time enable the team to better consider the architectural and other technical aspects when prioritizing user stories. Such recommendation is in line with research in the field of agile development that emphasizes direct communication and direct learning. ¹⁵
D2	Delayed synchronization of list of specifications in MS Excel	R2	The list of specifications that is currently managed in MS Excel should be moved to JIRA. This will integrate backlog with task planning and task completion administration. This recommendation is in line with research in application life cycle management that emphasizes requirements traceability. ⁶⁸
D3	Unsuccessful management pressure to raise performance	R3	Instead of pressure to stop undesirable behavior, we suggest management focuses on rewarding desirable behavior. Thus, we suggest to revamp the reward scheme and make a desired but still achievable percentage of tasks/bugs completed in time an important metric of remuneration. This recommendation is in line with research in the field of agile development where teams' ability to adapt the scope is emphasized. ³⁶
D4	Often exceeded estimated sprint tasks time limits in implementing new code and bug fixing	R4	To address this difficulty, we propose increasing transparency by using individual developer performance data mined from JIRA event logs to increase the efficiency of task distribution between the developers. The developers who are significantly more efficient at bug fixing should predominantly work on bug fixing, whereas the developers who are significantly more efficient at implementing new code should predominantly work on new code. Additionally, the first, the second, and the third improvement also partially address this difficulty. This recommendation increases transparency that is a corner stone of collaborative culture and is in line with research in the field of agile development. ⁶⁹

Abbreviation: SDM, software development methodology.

4.6 | Reflection workshop with company management

The identified difficulties and prepared recommendations were presented and discussed with the company management in order to assess their validity and usefulness. The management provided feedback as follows:

- The management strongly agreed with the first identified difficulty and the recommendation of the experts to give developers more autonomy and direct contact to the customer. They modified their requirements acquisition process so that it was performed by dedicated team members and not by the management.
- The management strongly agreed with the second identified difficulty and followed the recommendation to move the list of specifications to JIRA. This simplified the development process and reduced the process quality issues.
- The management strongly agreed with the third identified difficulty, but only partially followed the third recommendation. They did lessen the pressure on the developers but did not change the remuneration by linking it to performance. To lessen the pressure, they organized a yearly meet-up with core customers where developers themselves could estimate the time needed for development of requested features. This prevented the management to overpromise and overcommit to the core customers.
- The management strongly agreed with the fourth identified difficulty and followed the recommendation by providing the relevant performance data mined from JIRA event logs about the efficiency of task distribution between the developers to the team to improve their sprint planning.
- Management confirmed that this is a novel approach, and they could not specify any other similar approach. Existing SDM approaches (Scrum, Kanban, etc.) do not use process mining to evaluate SDM elements.
- Management confirmed that the proposed approach was useful and relevant in the context of SDM.

Overall, we found that management not only agreed with the identified difficulties but also found the results highly relevant to improve their process. As they stressed in the interview, despite the rich set of features provided by current supporting tools such as JIRA, there is still an unmet need for a higher level integrated reporting process that can give better insights on the overall status of the SDM.

5 | DISCUSSION

5.1 | Addressing research questions

We successfully addressed the three research questions posed in the paper and found that the proposed approach enables managers to gain additional insights into employees' performance (RQ1), because the information from software development tools logs allows for an improved performance management of individual developers by knowing their exact on-time sprint task/bug completion rates (Table 2). Next, we confirmed that the proposed approach delivers additional insights into project performance through monitoring the frequency of total tasks/bugs not completed in a single sprint (RQ2). Finally, we ascertained that the concurrent analysis of software development perception data and software development tools logs enables additional SDM improvement recommendations (RQ3). The first and second recommendation (R1 and R2 in Table 3) are based on perception analysis alone because log analysis does neither help us identify the first two key difficulties (D1 and D2 in Table 3) as important software development issues nor it can be used to improve the two recommendations (R1 and R2). However, the development of the third and fourth recommendations (R3 and R4 in Table 3) would not be possible without the concurrent analysis of perception and log data. R3 and R4 require software development tools logs to provide exact sprint task and bugs completion rates needed for improved individual developer performance management, whereas the perceptions provide the context about managers' satisfaction with specific performance levels.

5.2 | Contribution

Our case study showed several theoretical and practical implications of the proposed approach. The main theoretical contribution of our paper is a novel approach for evaluation of performance of SDM activities through concurrent analysis of stakeholder (managers and developers) perceptions and relevant software development tools logs. Its contribution is not primarily in specifying how managers and experts should develop insights and recommendations, but to present how combined sources of information (software development tools logs and perception data) will allow them to develop better recommendations and gain insights that otherwise would not be achievable. The approach combines the state of the art from the field of SDM evaluation (presented in Section 2.1) and the field of log data analysis (presented in Section 2.2) that were previously not connected. We hope that establishing this bridge enriches both fields and contributes to their further development.

We practically demonstrated how software development tools logs enable management to gain additional insights in the software development process regarding the performance of individual developers. Additionally, we showed how software development tools logs could be used to monitor the performance of an agile team by tracking the percentage and number of unfinished tasks after each sprint. It is important to know that these findings can only be gained by concurrently analyzing subjective qualitative SDM perceptions of developers and management, and quantitative software development tools logs data that importantly complement each other. On one hand, focusing only on log data lacks stakeholder context and is thus often difficult or even impossible to contextually correctly interpret. On the other hand, focusing only on stakeholder SDM perceptions can often lack important details that allow management to gain deeper understanding of specific issues because perceptions are typically collected on a more general level.

Management response that confirmed the validity of the proposed approach was added to the discussion. To ensure construct validity, we used multiple sources of information including surveys and interviews with key stakeholders, log data, and confirmation of the results with management. Internal validity was confirmed through two activities. The first activity was careful analysis of stakeholder perception and software development tools logs where we did not identify any conflicting information. The second activity was addressing rival explanations with management, which showed consistency between our interpretation and management views. This indicates strong consistency and internal validity of information and explanation. A typical central European software development SME using agile SDM was selected for the case study to ensure representative results and as much external reliability as possible. We believe that the approach can be scaled up and also be used in nonagile settings; however, it was only tested in an agile SME setting. To ensure reliability use, case study protocol was carefully followed.⁶⁷

5.3 | Threats to validity and other limitations

Four important limitations need to be considered. Firstly, there are limitations of external validity, namely, generalization. At this stage, we were able to conduct only a single case study in an agile setting. Although, we picked an Austrian SME that is representative for SMEs in the IT sector

in central Europe, additional case studies in multiple settings would help confirm the broader usefulness of the approach as well as strengthen its theoretical and practical implications. Secondly, the approach can only be used in enterprises in which software development tool logs are available. Thirdly, the study should be repeated in different cultural settings to see if other cultural contexts would importantly affect the usefulness of the approach. Lastly, due to variability of software development tools logs, it is not possible to define a universal step-by-step procedure for analysis of software development tools logs; however, this is often the case in the field of process mining.

Similarly, as other approaches in the field of process improvement,⁷⁰ our approach does not specify a step-by-step procedure with the aim to replace experts and managers and their ability to interpret data and develop recommendations. On the contrary, it empowers experts and managers with new information based on concurrent analysis. Moreover, it is impossible to formalize a procedure that would produce appropriate recommendation for each and every company because the context of companies can differ greatly. Thus, experts and managers need to be employed to interpret the information provided by our approach and produce the final insights and recommendations.

6 | CONCLUSION

The main contribution of the study is a novel approach for evaluation of software development process that concurrently considers stakeholder SDM perceptions and data from software development tools logs. In a case study, we confirmed the effectiveness of the proposed approach and identified its theoretical and practical contributions to the studied field. We successfully addressed the three research questions posed in the paper and discussed its validity and limitations.

Future work should expand the number of case studies in different cultural settings and test different sources of log data that would enable measuring not only developer performance, but also process and product quality. Furthermore, in the field of SDM perception analysis, additional stakeholders should be considered (business partners, customers, etc.). Finally, as the field of process mining is rapidly evolving, new methods and tools should be considered. Current automation trends in software development field generate ever increasing number of software development tools logs that will offer future researchers an opportunity to perform more comprehensive long-term analyses.

ACKNOWLEDGMENTS

The research by Saimir Bala and Jan Mendling was supported by the Einstein Foundation Berlin under Grant EPP-2019-524. Open Access funding enabled and organized by Projekt DEAL.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from third party. Restrictions apply to the availability of these data, which were used under license for this study. Data are available from the authors with the permission of third party.

ORCID

Damjan Vavpotič  <https://orcid.org/0000-0002-8039-7110>

Saimir Bala  <https://orcid.org/0000-0001-7179-1901>

Jan Mendling  <https://orcid.org/0000-0002-7260-524X>

Tomaž Hovelja  <https://orcid.org/0000-0002-3278-1433>

REFERENCES

- Vavpotič D, Robnik-Sikonja M, Hovelja T. Exploring the relations between net benefits of IT projects and CIOs' perception of quality of software development disciplines. *Bus Inf Syst Eng*. 2020;62(4):347-360. doi:10.1007/s12599-019-00612-4
- Kneuper R. *CMMI Improving Software and Systems Development Processes Using Capability Maturity Model Integration (CMMI-DEV)*. Santa Barbara, Calif: Rocky Nook; 2009.
- Laporte C, O'Connor R, García Paucar LH. The implementation of ISO/IEC 29110 software engineering standards and guides in very small entities. In *Evaluation of Novel Approaches to Software Engineering* (pp. 162-179). Springer International Publishing; 2016. doi:10.1007/978-3-319-30243-0_9
- Loon HV. *Process Assessment and ISO/IEC 15504: A Reference Book*. New York: Springer; 2007.
- Balasubramanian L, Mnkandla E. An evaluation to determine the extent and level of agile software development methodology adoption and implementation in the Botswana software development industry. In: Kumar V, Singh UG, Sudarsan SD, eds. *2016 Third International Conference on Advances in Computing, Communication and Engineering (ICACCE 2016)*. IEEE; 2016:320-325.
- Chan FK, Thong JY. Acceptance of agile methodologies: a critical review and conceptual framework. *Decis Support Syst*. 2009;46(4):803-814. doi:10.1016/j.dss.2008.11.009
- Vavpotič D, Hovelja T. Improving the evaluation of software development methodology adoption and its impact on enterprise performance. *Comput Sci Inf Syst*. 2012;9(1):165-187. doi:10.2298/CSIS110503072V
- Gill AQ, Henderson-Sellers B, Niazi M. Scaling for agility: a reference model for hybrid traditional-agile software development methodologies. *Inf Syst Front*. 2018;20(2):315-341. doi:10.1007/s10796-016-9672-8

9. Hovelja T, Vasilecas O, Vavpotič D. Exploring the influences of the use of elements comprising information system development methodologies on strategic business goals. *Technol Econ Dev Econ*. 2015;21(6):885-898. doi:10.3846/20294913.2015.1074130
10. Vavpotič D, Bajec M. An approach for concurrent evaluation of technical and social aspects of software development methodologies. *Information and Software Technology*. 2009;51(2):528-545. doi:10.1016/j.infsof.2008.06.001
11. Choetkiertikul M, Dam H, Tran T, Ghose A, Grundy J. Predicting delivery capability in iterative software development. *IEEE Trans Software Eng*. 2018; 44(6):551-573. doi:10.1109/TSE.2017.2693989
12. Mäntylä M, Adams B, Destefanis G, Graziotin D, Ortu M. Mining valence, arousal, and dominance: possibilities for detecting burnout and productivity? In: *Proceedings of the 13th International Conference on Mining Software Repositories (MSR '16)*. Austin, Texas: Association for Computing Machinery; 2016:247-258.
13. Ortu M, Murgia A, Tourani P, Tonelli R, Marchesi M, Adams B. The emotional side of software developers in JIRA. In: *Proceedings of the 13th International Conference on Mining Software Repositories, MSR*. Austin, Texas: ACM; 2016:480-483.
14. Ortu M, Destefanis G, Kassab M, Marchesi M. Measuring and understanding the effectiveness of JIRA developers communities. In: *6th IEEE/ACM International Workshop on Emerging Trends in Software Metrics, WETSoM*. Florence, Italy: IEEE Computer Society; 2015:3-10.
15. Annosi MC, Martini A, Brunetta F, Marchegiani L. Learning in an agile setting: a multilevel research study on the evolution of organizational routines. *J Bus Res*. 2020;110:554-566. doi:10.1016/j.jbusres.2018.05.011
16. Bass JM. Artefacts and agile method tailoring in large-scale offshore software development programmes. *Information and Software Technology*. 2016; 75:1-16. doi:10.1016/j.infsof.2016.03.001
17. Tuape M, Hasheela-Mufeti VT, Kayanda A, Porras J, Kasurinen J. Software engineering in small software companies: consolidating and integrating empirical literature into a process tool adoption framework. *IEEE Access*. 2021;9:130366-130388. doi:10.1109/ACCESS.2021.3113328
18. Zdravkovic J, Svee E-O, Giannoulis C. Capturing consumer preferences as requirements for software product lines. *Requirements Engineering*. 2015; 20(1):71-90. doi:10.1007/s00766-013-0187-2
19. Avison DE, Fitzgerald G. *Information systems development: methodologies, techniques and tools*. 4th ed. Maidenhead, Berkshire: McGraw-Hill; 2006.
20. Fitzgerald B. An empirical investigation into the adoption of systems development methodologies. *Inf Manage*. 1998;34(6):317-328. doi: 10.1016/S0378-7206(98)00072-X
21. Riemenschneider CK, Hardgrave BC, Davis FD. Explaining software developer acceptance of methodologies: a comparison of five theoretical models. *IEEE Trans Softw Eng*. 2002;28(12):1135-1145. doi:10.1109/TSE.2002.1158287
22. Aaen I. Software process improvement: blueprints versus recipes. *IEEE Software*. 2003;20(5):86-93. doi:10.1109/MS.2003.1231159
23. Fitzgerald B. Formalized systems development methodologies: a critical perspective. *Inf Syst J*. 1996;6(1):3-23. doi:10.1111/j.1365-2575.1996.tb00002.x
24. Huisman M, Iivari J. Deployment of systems development methodologies: perceptual congruence between IS managers and systems developers. *Inf Manage*. 2006;43(1):29-49. doi:10.1016/j.im.2005.01.005
25. Destefanis G, Ortu M, Counsell S, Swift S, Marchesi M, Tonelli R. Software development: do good manners matter? *PeerJ Comput Sci*. 2016;2:e73 doi: 10.7717/peerj-cs.73
26. Aboelmeleg MG. Predicting e-procurement adoption in a developing country: an empirical integration of technology acceptance model and theory of planned behaviour. *Ind Manag Data Syst*. 2010;110(3):392-414. doi:10.1108/02635571011030042
27. Venkatesh V, Davis FD. A theoretical extension of the technology acceptance model: four longitudinal field studies. *Manage Sci*. 2000;46(2):186-204. doi:10.1287/mnsc.46.2.186.11926
28. Wang W, Li X, Hsieh JP-A. The contingent effect of personal IT innovativeness and IT self-efficacy on innovative use of complex IT. *Behav Inf Technol*. 2013;32(11):1105-1124. doi:10.1080/0144929X.2011.566940
29. Rogers EM. *Diffusion of Innovations*. 5th ed. New York, NY [u.a.]: Free Press; 2003.
30. Gallivan MJ. The influence of software developers' creative style on their attitudes to and assimilation of a software process innovation. *Inf Manage*. 2003;40(5):443-465. doi:10.1016/S0378-7206(02)00039-3
31. Green GC, Hevner AR, Collins RW. The impacts of quality and productivity perceptions on the use of software process improvement innovations. *Information and Software Technology*. 2005;47(8):543-553. doi:10.1016/j.infsof.2004.10.004
32. Huisman M, Iivari J. The organisational deployment of systems development methodologies. In M. Kirikova, J. Grundspenkis, W. Wojtkowski, W. G. Wojtkowski, S. Wrycza, & J. Zupančič (Eds.), *Information Systems Development: Advances in Methodologies, Components, and Management* (pp. 87-100). Boston, MA: Springer US; doi:10.1007/978-1-4615-0167-1_8
33. Atkinson R. Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. *Int J Proj Manag*. 1999;17(6):337-342. doi:10.1016/S0263-7863(98)00069-6
34. Cooper RB, Zmud RW. Information technology implementation research: a technological diffusion approach. *Manage Sci*. 1990;36(2):123-139. doi: 10.1287/mnsc.36.2.123
35. Hoda R, Noble J, Marshall S. The impact of inadequate customer collaboration on self-organizing agile teams. *Information and Software Technology*. 2011;53(5):521-534. doi:10.1016/j.infsof.2010.10.009
36. Jorgensen M. Relationships between project size, agile practices, and successful software development results and analysis. *IEEE Software*. 2019;36(2): 39-43. doi:10.1109/MS.2018.2884863
37. Scott E, Milani F, Kilu E, Pfahl D. Enhancing agile software development in the banking sector—a comprehensive case study at LHV. *J Softw: Evol Process*. 2021;33(7): doi:10.1002/smr.2363
38. Karlsson F, Ågerfalk P. Exploring agile values in method configuration. *Eur J Inf Syst*. 2009;18(4):300-316. doi:10.1057/ejis.2009.20
39. Malinova M, Gross S, Mendling J. A study into the contingencies of process improvement methods. *Inf Syst*. 2022;104:101880 doi:10.1016/j.is.2021.101880
40. Ralyté J, Deneckère R, Rolland C. Towards a generic model for situational method engineering. *Proceedings of the 15th International Conference on Advanced Information Systems Engineering* (pp. 95-110). Berlin: Springer-Verlag; 2003. <http://dl.acm.org/citation.cfm?id=1758398.1758410>
41. Fontdevila D, Genero M, Oliveros A, Paez N. Evaluating the utility of the usability model for software development process and practice. In Franch, X and Mannisto, T and MartinezFernandez, S (Ed.), *Product-Focused Software Process Improvement, PROFES* 2019;11915:741-757. doi:10.1007/978-3-030-35333-9_{5}{7}

42. Dumas M, La Rosa M, Mendling J, Reijers HA. *Fundamentals of Business Process Management* (2nd ed.). Springer; 2018. doi:10.1007/978-3-662-56509-4.
43. van der Aalst WM. *Process Mining—Data Science in Action* (2nd ed.). Springer; 2016. doi:10.1007/978-3-662-49851-4.
44. Celonis. 2021. <https://www.celonis.com/>. Retrieved July 2021, from <https://www.celonis.com/>
45. Disco. 2021. <https://fluxicon.com/disco/>. Retrieved July 2021, from <https://fluxicon.com/disco/>
46. Apomore. 2021. <https://apomore.org/>. Retrieved July 2021, from <https://apomore.org/>
47. minit. 2021. <https://www.minit.io/>. Retrieved July 2021, from <https://www.minit.io/>
48. Labs L. 2021. <https://lanalabs.com/en/>. Retrieved July 2021, from <https://lanalabs.com/en/>
49. Viner D, Stierle M, Matzner M. A process mining software comparison. *CoRR, abs/2007.14038*. 2020. Retrieved July 2021, from <https://arxiv.org/abs/2007.14038>
50. Bala S, Cabanillas C, Mendling J, Rogge-Solti A, Polleres A. *Mining Project-Oriented Business Processes*. BPM Conference. Springer; 2015. doi:10.1007/978-3-319-23063-4_28.
51. Bala S, Mendling J. Monitoring the software development process with process mining. In: *Business Modeling and Software Design - 8th International Symposium, BMSD 2018*. Vienna, Austria: Springer; 2018.
52. Bala S, Revoredo K, AR Goncalves JC, Araujo Baiao F, Mendling J, Santoro FM. Uncovering the hidden co-evolution in the work history of software projects. In: *Business Process Management—15th International Conference, BPM 2017*. Barcelona, Spain: Springer; 2017.
53. Bala S, Kneringer P, Mendling J. Discovering activities in software development processes. In: *PoEM (Forum)*. Vol. 2793. Riga: CEUR-WS.org; 2020: 54-63.
54. Marquez R, Mira da Silva M, Ferreira DR. Assessing agile software development processes with process mining: a case study. *IEEE 20th Conference on Business Informatics (CBI)*. IEEE; 2018.
55. Grisold T, Wurm B, Mendling J, vom Brocke J. Using process mining to support theorizing about change in organizations. In: *53rd Hawaii International Conference on System Sciences, HICSS 2020*. Maui: ScholarSpace; 2020:1-10.
56. Gross S, Malinova M, Mendling J. Navigating through the maze of business process change methods. In: *52nd Hawaii International Conference on System Sciences, HICSS 2019*. Maui, Hawaii: ScholarSpace; 2019:1-10.
57. Revoredo K, Djurica D, Mendling J. A study into the practice of reporting software engineering experiments. *Empir Softw Eng*. 2021;26(6): doi:10.1007/s10664-021-10007-3
58. Hevner AR, March ST, Park J, Ram S. Design science in information systems research. *MIS Quarterly*. 2004;28(1):75-105. doi:10.2307/25148625
59. Henderson-Sellers B, Ralyté J, Ågerfalk PJ, Rossi M. *Situational Method Engineering*. Springer; 2014. doi:10.1007/978-3-642-41467-1.
60. van Eck ML, Lu X, Leemans SJ, van der Aalst WM. PM2: a process mining project methodology. In: *Advanced Information Systems Engineering, CAiSE 2015*. Stockholm, Sweden: Springer, Cham; 2015:297-313.
61. Leemans SJ, Fahland D, van der Aalst W. Process and deviation exploration with inductive visual miner. *BPM (Demos)*. 2014;1295(8):
62. van der Aalst WM, Weijters T, Maruster L. Workflow mining: discovering process models from event logs. *IEEE Trans Knowl Data Eng*. 2004;16(9): 1128-1142. doi:10.1109/TKDE.2004.47
63. Verbeek H, Buijs JC, van Dongen BF, van der Aalst WM. XES, XESame, and ProM 6. In: *International Conference on Advanced Information Systems Engineering, CAiSE Forum*. Hammamet, Tunisia: Springer; 2010:60-75.
64. Hamdy A, Ezzat G. Deep mining of open source software bug repositories. *Int J Comput Appl*. 2020;0:1-9. doi:10.1080/1206212X.2020.1855705
65. Takeuchi H, Nonaka I. The new new product development game. *Harv Bus Rev*. 1986;137-146.
66. Runeson P, Höst M. Guidelines for conducting and reporting case study research in software engineering. *Empir Softw Eng*. 2008;
67. Yin RK. *Case Study Research: Design and Methods (Applied Social Research Methods)*. 4th ed. Sage Publications; 2009.
68. Akman S, Ozmut M, Aydin B, Gokturk S. Experience report: implementing requirement traceability throughout the software development life cycle. *J Softw: Evol Process*. 2016;28(11):950-954. doi:10.1002/smr.1824
69. Luz WP, Pinto G, Bonifacio R. Adopting DevOps in the real world: a theory, a model, and a case study. *J Syst Softw*. 2019;157: doi:10.1016/j.jss.2019.07.083
70. Zellner G. A structured evaluation of business process improvement approaches. *Bus Process Manag J*. 2011;17(2):203-237. doi:10.1108/14637151111122329

How to cite this article: Vavpotič D, Bala S, Mendling J, Hovelja T. Software Process Evaluation from User Perceptions and Log Data. *J Softw Evol Proc*. 2022;34(4):e2438. doi:10.1002/smr.2438