Assessing Agile Software Development Processes with Process Mining: A Case Study

Rita Marques, Miguel Mira da Silva and Diogo R. Ferreira
Instituto Superior Técnico (IST)
Lisbon, Portugal
Email: {rita.marques, mms, diogo.ferreira}@tecnico.ulisboa.pt

***Abstract*—Software development is a growing industry with increasingly more complex projects. Agile methodologies, such as Scrum, have emerged in part to address this challenge. Agile focuses on a set of best practices rather than a standardized process, which makes it difficult to assess whether it is being properly implemented in an organization. In this work, we analyze the feasibility of using process mining to evaluate the implementation of Scrum practices based on event logs collected from a case-handling system. A case study was conducted in an IT organization that uses Jira Software to manage its projects. With process mining, it was possible to extract the workflow behaviour in two different projects. While Scrum standards like role distribution could be discovered, other practices like customer collaboration could not be detected in the log. This case study provided important insights regarding the application of process mining in case-handling systems and Scrum processes.**

***Keywords*—Agile software development, Scrum methodology, process mining, case-handling systems**

I. Introduction

Software development companies are adopting new tools and methodologies to cope with the increasing complexity of their projects and end-user demands [1], [2]. Approaches combining agile methodologies (like Scrum [3]) and collaboration tools (like case-handling systems [4]) have emerged as a flexible solution to address these challenges. However, such flexibility also makes it hard to assess whether an agile methodology is being correctly implemented in practice.

Process mining [5] provides a set of techniques to automatically extract process behavior from event logs. This approach has been successfully applied in several fields, such as healthcare [6], IT service management [7], [8], software development [9], etc. Process mining techniques can also help organizations evaluate which agile practices are being implemented, based on event logs collected from case-handling systems. The present work focuses on Scrum, since it is the most widely adopted agile methodology. In particular, we try to identify which Scrum practices can be discovered using process mining techniques.

Specifically, this work aims at answering the following research questions:

- RQ1: Which Scrum recommended practices can be discovered using process mining techniques?
- RQ2: Which issues should be taken into account when extracting and analyzing event logs with unstructured behavior from Scrum processes?

A case study was conducted in an IT organization that uses Scrum and Jira Software[1] to manage its projects. The case study illustrates how process mining techniques can be applied in the context of case-handling systems and the Scrum methodology. It shows how far one can go in using process mining to assess the implementation of agile methodologies. This will be of interest not only to researchers in this field, but also to organizations who are looking forward to improve their software development processes.

The structure of the paper is as follows. After an overview of related work, we present the case study that has been carried out through several stages, including planning, data extraction and processing, mining and analysis activities, and results evaluation. The approach follows the PM^2 process mining methodology as proposed in [10]. After discussing the results, the main conclusions are presented.

II. Related work

Unlike other software development methodologies, *agile* is not a standardized process with clearly defined steps [11]. Instead, it is a methodology that focuses on a set of best practices, which advocate that people and their relations are more important than following specific processes or tools [12].

Scrum is the most widely adopted agile methodology [13], where work is visible to the entire team, so that every developer always knows what is happening in the project and can adjust their practices to achieve the desired goals [3]. Scrum teams should be cross-functional, and self-organizing.

When a Scrum project starts, the *product owner* creates a *product backlog* with *user stories* covering all system requirements, and sorts them by priority. A *sprint* starts with the *sprint planning* meeting, where user stories with high priority are selected to build the *sprint backlog*. The *Scrum master* is the team leader, responsible for ensuring that Scrum is understood and that developers are adopting Scrum practices.

Stories are assigned to each member of the *development team*, and they may be further broken down into tasks. During the sprint, there is a *daily meeting* that is conducted to assess the project status and, at the end of the sprint, an *increment* of the product must be completed. The sprint finishes with *sprint review*, where the increment is presented and discussed,

[1] https://www.atlassian.com/software/jira

and *sprint retrospective* meetings, were the lessons learned are discussed to improve the following sprint [3].

Even though these practices can be adapted to the needs of a particular organization, a good Scrum implementation will comply with most of these practices and principles.

In summary, it can be said that Scrum includes a set of roles (*product owner*, *Scrum master* and *development team*), ceremonies (*sprint*, *sprint planning*, *daily meeting*, *sprint review*, and *sprint retrospective*) and artifacts (*product backlog*, *sprint backlog*, and *increment*). All of these elements interact during the software development process. In essence, the main goal of this work is to find out how much of this interaction can be discovered through process mining.

A. Process Mining

Process mining consists in extracting process behavior from event logs. This approach can be extremely useful to organizations who depend on information systems to manage and execute their operational processes [14]. Based on a large amount of run-time event data recorded by those systems, process mining provides the means to discover the sequence of activities and user collaborations in a business process.

In an event log, each event contains information about the case (i.e. the process instance), activity (a step in the process), the resource (i.e. the user who performed the activity), and the timestamp [5]. Besides these fields, other case and event attributes may be included as well.

It is possible to analyze the event log from different perspectives, namely: the control-flow perspective, which focuses on the activities in order to find a characterization of the possible paths; the organizational perspective, which focuses on the resources to discover their interactions and collaborations during the process; and the performance perspective, which focuses on the time it takes to perform each activity [5].

Several techniques can be applied to event logs, such as the α-miner [15], the inductive miner [16], the heuristics miner [17], the fuzzy miner [18], and algorithms to discover social networks [19], for example. These techniques are supported by research tools and commercial tools, including ProM [20] and Disco [21].

B. Process mining and software development

The application of process mining in the field of software development has already been investigated by several authors, namely:

- Poncin et al. [22] propose the use of process mining techniques to mine information extracted from different software repositories. For this purpose, they developed an application to combine information from various software repositories and evaluated it in case studies from different domains. They conclude that process mining can provide meaningful insights into software development processes.
- Bala et al. [23] also studied how to mine software repositories in order to understand software projects. Using event logs from version control systems, they generate Gantt charts for project managers to visualize what was done in the project.
- Rubin et al. [24] discuss the idea of using information stored in software configuration management systems to construct explicit software development process models, which are often implicit and chaotic. They conclude that it is possible to obtain process models from real-world software projects, using process mining and the ProM tool in particular.
- Rubin et al. [25] suggest the embedding of process mining into the agile development lifecycle to derive conclusions regarding the real user and system runtime behavior. They argue that agile environments align with process mining preconditions, since software produced according to this methodology is constantly delivered and used by meaningful stakeholders, which leads to the continuous recording of event logs.
- Lemos et al. [9] applied process mining techniques to measure conformance checking between event logs and a well-defined software development process. Furthermore, they discuss the use of process mining to improve the maturity level of software engineering organizations, as defined by the Capability Maturity Model Integration (CMMI).

In general, most of these works focus on extracting the software development process rather than checking its underlying methodology. The work by Lemos et al. is an exception since they use process mining to assess whether the process conforms to its formal specification. However, this formal specification is given as a structured process, which does not apply to agile.

C. Process mining methodology

In the literature, several methodologies regarding the application of process mining have been proposed, including the L^* life-cycle model [5], the Process Diagnostics Method [26], the Process Comparison Methodology [27], and the PM^2 process mining methodology [10]. In this work we adopt the PM^2 methodology, since it is general enough to support the analysis of both structured and unstructured processes.

For later reference, the PM^2 methodology comprises the following stages:

1) *Planning*: setting up the project, including the choice of which information system to extract information from.
2) *Extraction*: deciding what to extract (event data or process models, period of the events, data attributes, etc.) and performing the extraction.
3) *Data Processing*: filtering the relevant activities and creating views over specific parts of the dataset that are interesting.
4) *Mining and Analysis*: applying the process mining techniques.
5) *Evaluation*: establishing the relation between findings and improvement ideas, which must be validated with stakeholders involved in the process.

6) *Process Improvement and Support*: modification of the actual process execution, based on the improvement ideas.

D. Case-handling systems

Case-handling systems are a more flexible alternative to workflow systems, which require participants to follow a structured process model [4], [28]. Instead of prescribing what to do, case-handling systems often describe what is not allowed to do, enabling less structured processes, where process execution can greatly vary, depending on the case and on who is working on it.

In case-handling systems, all the information related to a case can be made visible to all workers at all times, providing an effective overview of everything that has been done or is yet to be done to complete the case. This can help in reducing the effect of *context tunneling* [29], a situation where workers are isolated in their own activities and lack a big picture of the case.

A popular case-handling system is Jira Software,[2] a tool that supports the entire software development process, including planning, releasing, tracking, and reporting. The system is built on the concept of *issue*, which represents a problem that needs to be solved, after being classified according to its type (e.g. bug, improvement, or user story). Typically, an organization defines the states that an issue is allowed to go through. Issues are composed by a set of data fields, including priority (importance of an issue), resolution (the way an issue was closed), and reporter (the worker that created the issue).

In Jira Software, the *Scrum master* can create new issues in the backlog and associate them to a specific sprint. Developers are assigned issues to work on, according to the distribution across the team.

III. Case study

In this work, we analyzed two projects (referred to as ProjectX and ProjectY) in an IT organization providing software products and services. The company manages its software development processes through Jira Software and Scrum. By using process mining to understand the user behavior in the two projects, it is possible to analyze how Scrum is being implemented in this organization.

As stated before, to conduct this process mining analysis, we adopted the PM^2 methodology. The next subsections describe the work performed in each stage, excluding the final stage (process improvement and support), which was not applied.

A. Planning

The main goal of this case study was to analyze the implementation of Scrum practices in ProjectX and ProjectY, aiming at identifying which Scrum practices can be discovered by process mining techniques. For this purpose, data was extracted from the case-handling system adopted by the organization (Jira Software), even though Scrum is not entirely supported by that system. In particular, no information is recorded regarding Scrum meetings and roles. Nevertheless, it is possible to verify some Scrum best practices, such as having every issue assigned to a resource and not having any reopened issues. In addition, we intended to investigate if other practices could be detected during the analysis, such as the occurrence of Scrum ceremonies.

B. Extraction

To extract data from the system, a Web-based application using the Jira REST API[3] was developed. Basically, the application lists the available projects in a given Jira instance by making an authenticated HTTP request to the server. Then, the projects of interest are filtered to create a JSON with general information about each project and a detailed list of issues.

The application subsequently iterates through the issues to retrieve specific information about them. In particular, we retrieve the ID, type, priority, resolution, and reporter for each issue. We also need the status of an issue, and all the states that the issue went through, since the execution of an activity corresponds to a change in status.

To access the status changes, the assignees who performed them, and the time when they occurred, the history section of each issue had to be analyzed. This was achieved by making an HTTP request to the access link for each issue, and transforming the returned XML into a CSV format using a XSLT template. In the resulting event log, each entry contains the case id, task, assignee, timestamp, type, priority, resolution, and reporter.

C. Data Processing

During data processing, the resource names were anonymized (as *R1*, *R2*, etc.), and the event log was split in two parts (one for each project), as the goal was to first analyze them separately and then compare results. Open issues (i.e. those that have not been completed yet) were filtered out from the event log. The conditions applied for this filtering were: having the resolution state *Unresolved*, and having an end status other than *Resolved* or *Closed*.

D. Mining and Analysis

ProjectX comprised 795 issues, with 6011 events performed between Dec. 2012 and March 2017 by 50 assignees. Issues were created by 46 reporters, and the process has 176 variants. There are 15 different types of activities. *Reopened* events correspond to 1.55% of the log.

After importing the event log into ProM, it was analyzed with the *dotted chart* [30], which displays the sequence of events for each issue over time. As shown in Fig. 1, each event is represented by a single dot, and a color code is used to indicate the event type. This technique provides a way to visualize all events at once and, if needed, it is possible to zoom in to investigate particular patterns. Some interesting patterns have been highlighted in Fig. 1.

[2]https://www.atlassian.com/software/jira/features

[3]https://docs.atlassian.com/jira/REST/server/

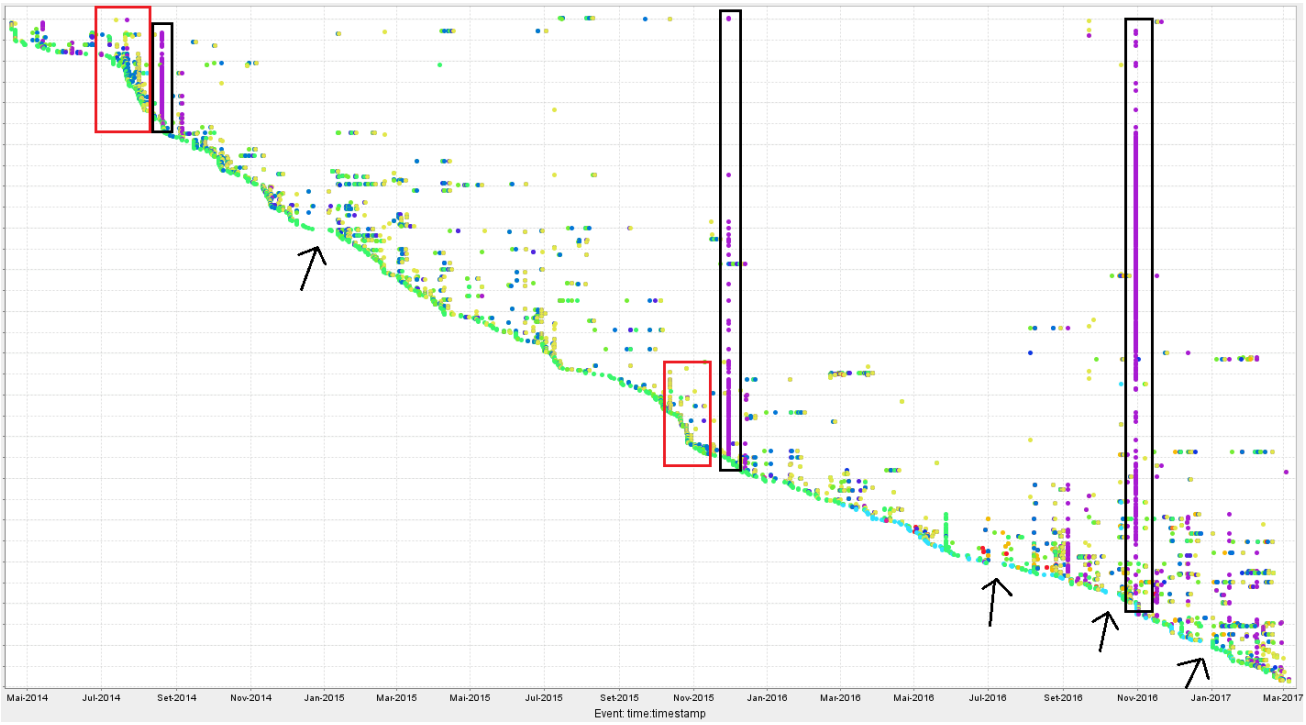


Fig. 1: Dotted chart for ProjectX, generated in ProM. The red rectangles highlight increases in the arrival rate, black rectangles represent areas where a large number of issues were closed, and the black arrows point to gaps in the arrival rate.

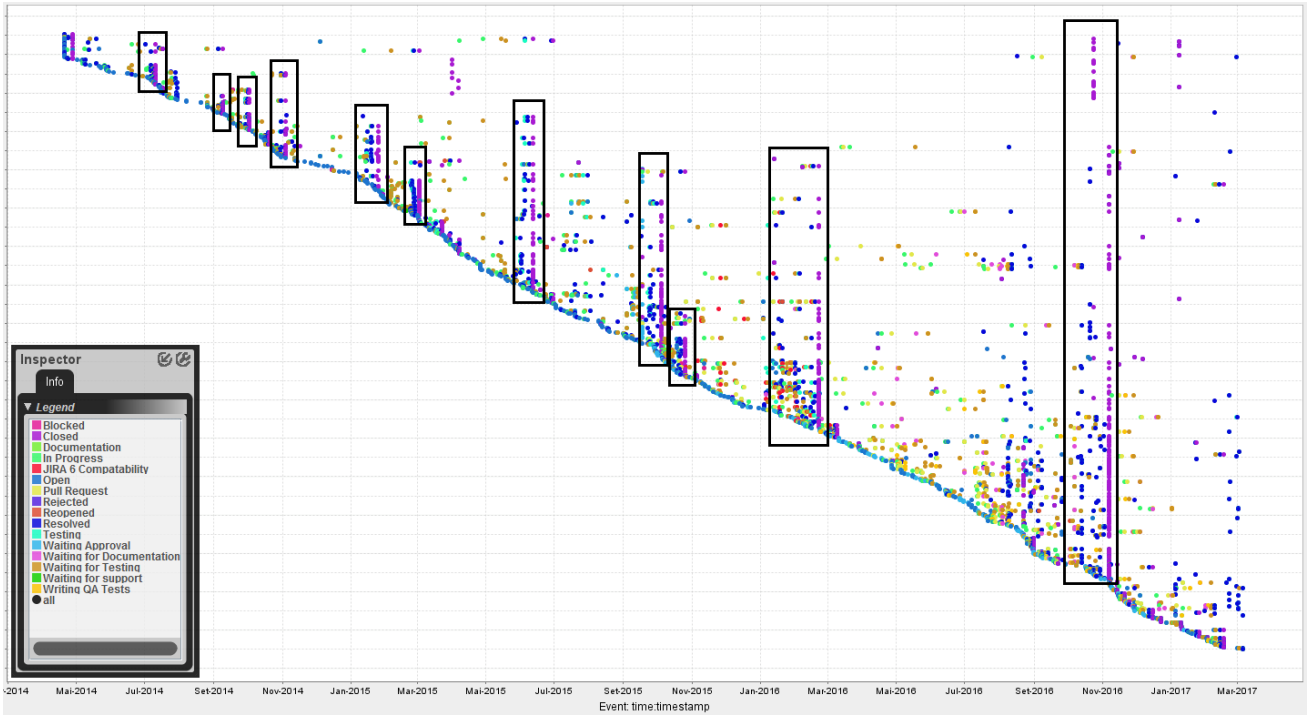


Fig. 2: Dotted chart for ProjectY , generated in ProM. The black rectangles represent areas where a lot of issues were closed.

In particular, the arrival rate is not constant: the areas delimited with a red rectangle show increases in the number of issues created (around July and Oct. 2014), while the black arrows denote gaps in process activity (end of Dec. 2014 / beginning of Jan. 2015; July 2016; Oct. 2016; end of Dec. 2016 / beginning of Jan. 2017).

Furthermore, there were three moments when a lot of issues were closed by the same assignee (illustrated by the black rectangles): Aug. 2014 (closed by *R1*), Dec. 2015 and Nov. 2016 (closed by *R2*). Other variants of the dotted chart show that the case duration has diminished through time, and that more issues are closed in the beginning of the week and at the beginning of the day.

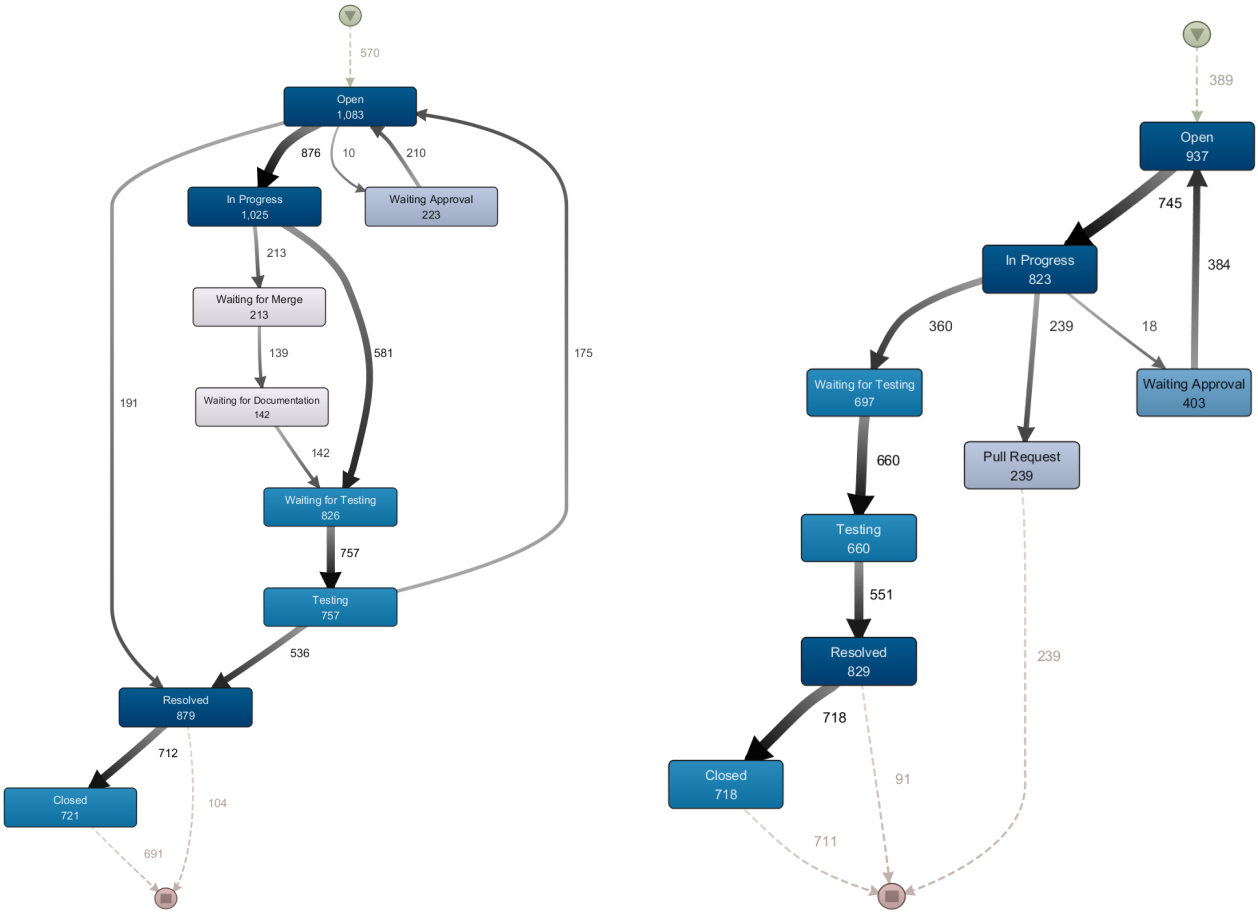
For ProjectY, a total of 82 issues and 5795 events were recorded between April 2014 and March 2017, performed by 69 assignees and created by 97 reporters. There are 16 different types, and 0.47% of the log correspond to *Reopened* events. The process presents 165 variants.

The arrival rate for ProjectY is more constant than ProjectX, as one can see in the dotted chart of Fig. 2. Except for a period in Aug. 2014, there are no gaps in process activity. There are periods where a lot of issues are closed, preceded by a batch of resolve activities. Resolving activities are mostly performed by *R4*, and closing activities by *R72*. Again, the analysis showed that case duration has diminished through time, and that more issues are closed in the beginning of the week, as observed in ProjectX.

Both event logs were analyzed according to different perspectives, namely:

1) Case perspective: In ProjectX, more than half of the issues are of type *Bug*, which is coherent with the fact that most issues are resolved as *Fixed* (93.78%). Regarding priority, most issues are considered *Major* (94.81%). These values are similar to those of ProjectY, where most issues are of type *Bug* (49.84%) and *Story* (37.4%) are resolved as *Fixed* (93.74%) and are considered *Major* (92.36%).

2) Control-flow perspective: Several algorithms were used to generate a process model, namely the α-miner, the heuristics-miner, the inductive miner, and the fuzzy miner. In general, the heuristics- and α-miner generated spaghetti models. The inductive miner produced a simpler model, but it had some misconceptions regarding the defined workflow (e.g., *Resolved* and *Closed* appearing in parallel when they are performed sequentially). Due to these issues, we turned to



(a) ProjectX (25% activities, 20% paths)

(b) ProjectY (20% activities, 0% paths)

Fig. 3: Process models generated by Disco.

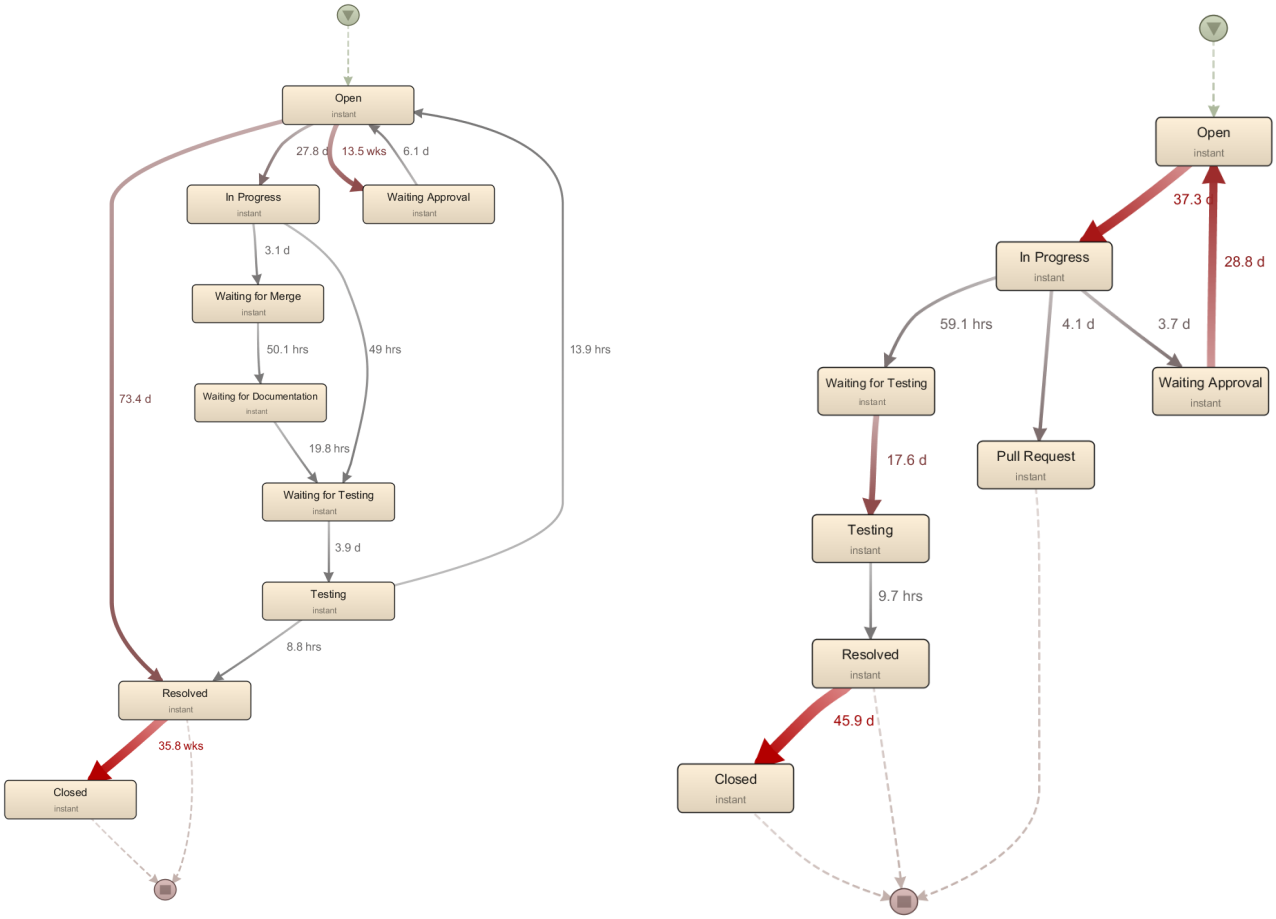
Fig. 4: A case variant of ProjectX, illustrating a pattern visible in both projects.

the fuzzy miner. Fig. 3 shows that, in both projects, the main path taken by an issue is: *Open* → *In Progress* → *Waiting for Testing* → *Testing* → *Resolved* → *Closed*.

Afterwards, some case variants were studied, which revealed another common pattern in both projects: a loop of four activities before the issue is eventually *Resolved* and *Closed*. Fig. 4 shows an example. The sequence starts with the activities *Open* → *In Progress* → *Waiting for Testing* → *Testing*, which are then repeated several times before the issue is *Resolved* and *Closed*. At first sight, this could point to the agile nature of the process, which may require multiple attempts until a user story is successfully implemented. However, this is not the case, as we will see later.

3) Performance perspective: Performance was analyzed based on the models displayed in Fig. 5, which show the time elapsed between states. In both projects, most transitions take some hours or even a few days to complete. In ProjectX, the transitions *Open* → *Waiting Approval* and *Open* → *Resolved* last on average 13.5 weeks (three months) and 73.4 days (two and a half months), respectively. However, the main bottleneck is the transition *Resolved* → *Closed*, which takes on average 35.8 weeks to complete (more than eight months). In ProjectY, although the transition *Waiting for Testing* → *Testing* takes 17.6 days (two and a half weeks) to complete, the major bottlenecks are *Waiting Approval* → *Open* (28.8 days, almost a month), *Open* → *In Progress* (37.3 days, more than a month), and *Resolved* → *Closed* (45.9 days, one and a half months), which is also a bottleneck transition in ProjectX.

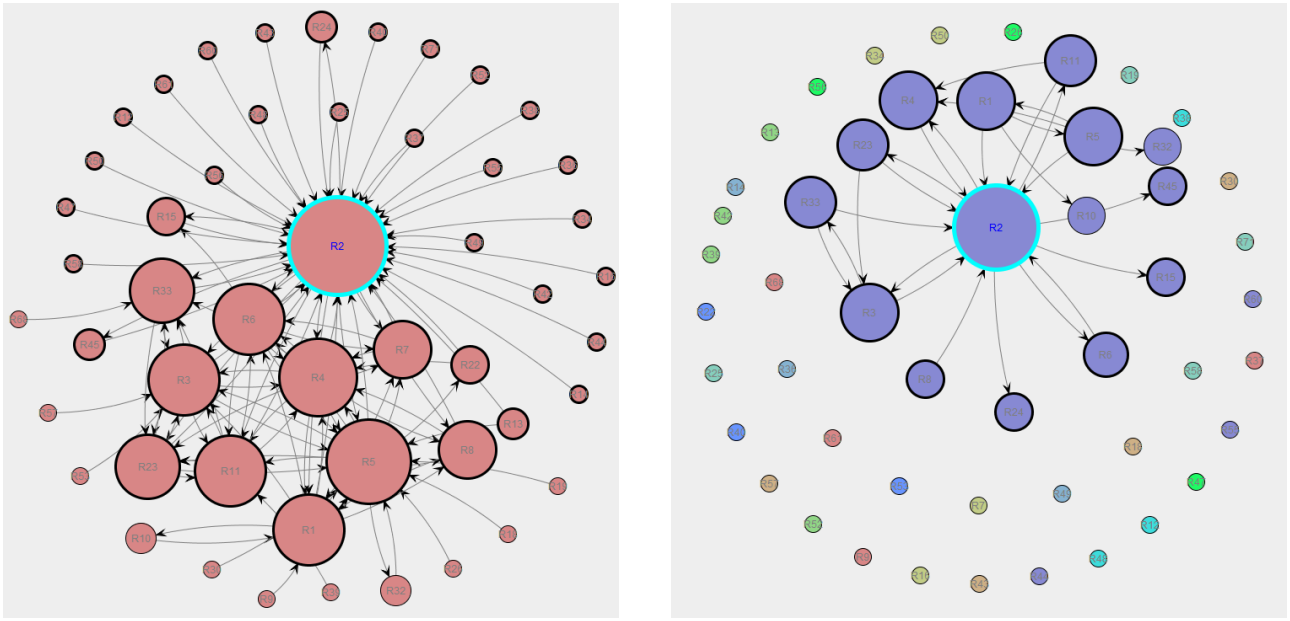
4) Organizational perspective: There are 50 assignees and 46 reporters in ProjectX. The assignees *R2* (36.12%), *R4* (15.57%), and *R5* (12.68%) are the most active, followed by *R3*, *R6*, *R1*, *R11*, *R23* and *R33*, each responsible for 1% to



(a) ProjectX

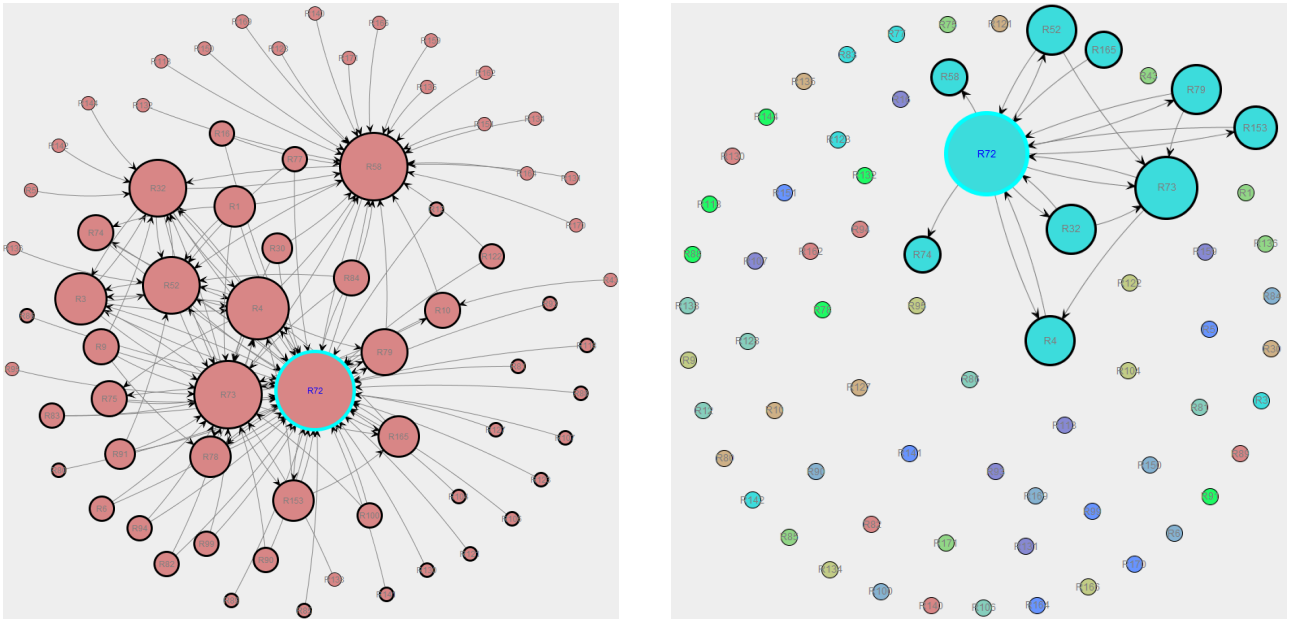
(b) ProjectY

Fig. 5: Performance perspective for both projects, based on mean duration.



(a) Handover of Work (b) Sub-Contracting

Fig. 6: Social networks for ProjectX. The central assignee R2 is selected (blue outline).



(a) Handover of Work (b) Sub-Contracting

Fig. 7: Social network for ProjectY. The central assignee R72 is selected (blue outline).

10% of the activities. Reporters *R2* (32.07%), *R4* (22.26%), and *R1* (12.83%) are the most active, followed by *R6*, *R3*, and *R11*, each responsible for creating between 1% and 6% of the issues. Other assignees and reporters are responsible for less than 1% of the activities.

By analyzing the *handover of work* [19] for ProjectX, which is displayed in Fig. 6a, it is apparent that *R2* is the main receiver of work. The remaining active assignees also exchange a lot of work between them, and receive work from less active assignees, who never receive any work. From the sub-contracting network in Fig. 6b it is possible to infer that active assignees subcontract in both directions, and less active assignees do not subcontract at all. Overall, *R2* has a central role in ProjectX, by working with almost everyone else.

In ProjectY, there are 69 assignees and 97 reporters. Assignee *R72* (42.86%) is the most active, followed by *R4* (19.67%) and *R73* (14.46%). Along with *R32*, *R52*, *R58*, and *R3*, these assignees are responsible for more than 1% of the activities. *R72* is the main reporter (43.35%), followed by *R73* (17.74%). *R32*, *R52*, *R4*, *R1*, *R58*, *R9*, *R84*, and *R79* are the other reporters with more than 1% of activity. Other assignees and reporters are responsible for less than 1% of the activities.

The social networks generated for ProjectY, presented in Fig. 7, are almost identical in structure with those of ProjectX. *R72* has the central role in ProjectY.

There are three resources common to both processes: *R1*, mainly contributing as a reporter; *R3*, mainly contributing as an assignee; and *R4*, contributing as both reporter and assignee. The organizational structure for both projects seems to be similar. One assignee has a central role in the process. A group of other assignees also work a lot together. Finally, a group of less active assignees only contribute to one or two cases, passing work to the more active group.

E. Evaluation

After analyzing the results, these were evaluated in a validation session with a project manager linked to both projects.

Due to the small team size of ProjectX, when workers go on vacation the project becomes idle, which justifies the gaps in process activity. There are less idle periods in ProjectY because its team is larger. Nevertheless, both teams have been growing to the size advocated by Scrum (three to nine people), which led to the reduction in case duration on both projects.

As observed, most issues are closed on Monday mornings in both projects. This is explained by the fact that this task is performed during Scrum review meetings, which take place in that period for both projects. Hence, based on this pattern it was possible to determine the timing of these meetings.

Also, while validating the results with the project manager, the following Scrum roles were identified based on the organizational perspective analysis:

- the Scrum master is the person with the central role and more activity;
- the development team is the small group of active assignees that mostly work together;
- the product owners are the people with very little activity, only opening issues.

In addition, analysis of the dotted chart revealed that testers (which is not a Scrum-related role) resolve issues, and Scrum masters close issues. Two of the three assignees that are common to both projects compose the testing team, which is shared between projects. This happens because these teams are task-specific and not cross-functional as advocated by Scrum.

Team self-organization could not be assessed with process mining tecniques. However, the project manager revealed that the organization does not follow this Scrum principle, and that Scrum masters are responsible for assigning issues to workers. Yet, all issues were assigned to some person, as advocated by Scrum.

Some bottlenecks were found in the performance perspective. In both projects, issues are only closed before launching a release into production, which is a task performed by Scrum masters. This explains the bottlenecks in the transition *Resolved* $\rightarrow$ *Closed*. In ProjectY, stories are only tested near the release date, which justifies the bottleneck in the transition *Waiting for Testing* $\rightarrow$ *Testing*. Nevertheless, this behavior does not comply with Scrum, since stories should be completely finished by the end of a sprint. Remaining bottlenecks are mostly related to the transition from the product backlog to a sprint backlog.

The fact that issues are closed only before launching a release into production explains the periods where a lot of issues are closed, highlighted with the black rectangles in Fig. 1 and Fig. 2: each period correspond to the launch of a new release. ProjectY has more releases than ProjectX because its team is larger, meaning that the team is able to finish work faster. In contrast to ProjectY, where stories are only tested near the release date, the product is continuously tested in ProjectX. However, both teams conduct exhaustive tests near the release date, thus explaining the red rectangles in Fig. 1, which correspond to two ProjectX releases where a lot of bugs were created.

Issues follow the same main path in both projects. Despite the path being consistent with a Scrum approach, the loop pattern found in Fig. 4, expressing the iterations over the implementation of an issue, does not represent an agile behavior. Instead, it shows that even though only a small portion of issues are being explicitly reopened, they are not being correctly developed (as they are frequently sent for testing with many bugs). As the loop pattern is repeated too many times, it might be leading to bottlenecks in the process. This was a valuable information for the project manager, who was not aware of this phenomenon.

Overall, despite the similarities in the way both projects are managed and organized, there are differences regarding the respective Scrum implementation.

IV. Discussion

From this case study, it was possible to derive useful insights regarding the use of process mining to evaluate the implemen-

tation of Scrum practices based on event logs extracted from a case-handling system.

With regard to the first research question (*Which Scrum practices can be discovered using process mining techniques?*), we were able to conclude the following:

1) It is possible to check whether all issues are prioritized and assigned to some resource.
2) It is possible to identify Scrum roles, even though they are not recorded in the case-handling system. With process mining, we were able to identify these roles by analyzing the organizational perspective. This revealed that all Scrum roles are implemented.
3) It is possible to assess the cross-functionality of the team based on patterns in the dotted chart. By validating the results with the project manager, we discovered that only testers resolve issues, which in turn are closed only by Scrum masters. This revealed that teams do not share responsibilities.
4) It is hard to evaluate if a team is self-organized based on the information contained in an event log.
5) One of the principles of the agile methodology is that customers and teams should continuously collaborate throughout the project. With process mining, it was possible to see that customers (with the product owner role) are opening issues, but no evidence was found regarding their ongoing collaboration with the team.
6) The analysis of Sprint review meetings can be based on the dotted chart, where it is possible to detect the timing when issues are resolved (a task performed during this meeting). However, no evidence was found regarding other types of Scrum meetings.
7) No insights could be derived regarding Scrum artifacts, and the event log did not allow to determine if stories are closed by the end of the sprint.

With regard to the second research question (*Which issues should be taken into account when extracting and analyzing event logs with unstructured behavior from Scrum processes?*), we discuss below a set of lessons learned about using process mining analysis to assess and improve Scrum practices:

1) The analyst should explore the case-handling system to understand which data can be exported and how. Because Scrum is not a structured process, the event data to be considered should go beyond the usual case id/task/resource/timestamp to include all attributes that can be useful regarding Scrum practices in an organization.
2) During mining and analysis, the analyst should examine the results with close attention, since important insights can arise unexpectedly. For example, in this case study we discovered the timing of Sprint review meetings based on a dotted chart pattern, which was not expected beforehand.
3) Discussing the results and findings with someone from the organization is a crucial step. In this case study, validation with the project manager provided a better understanding of how Scrum is implemented in the organization. Not only was this helpful for our research team, but also for the organization itself, as the study revealed some problems in the Scrum implementation that can now be internally addressed.
4) By using event data extracted from a case-handling system, some Scrum practices can be directly verified, like the fact that all issues are prioritized and assigned to some resource. Nevertheless, most insights arise from a deeper analysis of the results.
5) Some features of the control-flow model, for example the presence of loops, may point to issues that deserve further investigation. In the case study, loops detected during analysis showed that projects can present several iteratively reworked issues, which is not compliant with the Scrum standard, despite the fact that the fraction of reopened issues is small. The number of times that this loop pattern repeats varies, which was confirmed by manually verifying all case variants. A process mining technique that could automatically detect these loops and present them in a more business-friendly way would allow an easier identification of these problems.
6) Analyzing event data from several projects can reveal if the implementation of Scrum practices is consistent across the organization, or if it is specific to a single project. While being based on the analysis of an event log extracted from a particular system (Jira Software), this can be generalized to other case-handling systems as well, as long as the event log data can be extracted.
7) A Scrum implementation can be assessed by verifying the presence (or absence) of advocated practices. This is what we did in the case study. However, the degree of implementation of Scrum practices should also be addressed, but this is more difficult to quantify based on an analysis of the event log.
8) In general, when analyzing agile processes, the meaningful output of process mining is the verification of best practices, and not the process behavior itself. However, these practices are more abstract and harder to analyze than structured business processes, and require knowledge of Scrum and of the inner workings of the case-handling system. The organizational perspective becomes more important than the control-flow perspective, as the best practices advocate that the focus should be on people, rather than on process execution. Thus, it

becomes important to analyze the relationship between workers and their role in the project to derive meaningful conclusions about how to improve their effectiveness.

V. Conclusion

Agile methodologies provide the means to address the challenges faced by the software development industry, but their implementation is hard to assess from the events recorded in a case-handling system. Process mining can play a key role in such analysis by extracting process models and user behavior from those events.

In this work, a case study was conducted in an IT organization, where process mining was applied to extract information from two software projects that follow the Scrum methodology. The lessons learned in this case study can be useful when conducting similar analysis in other organizations and projects, especially (but not limited to) those supported by Jira Software. Moreover, some challenges were identified regarding the application of process mining to study software development processes based on data from case-handling systems. In our view, the combination of these three contexts should be further investigated.

A more immediate goal is to improve the Web application to extract more data from issues. This could help in discovering some of the practices that were not identified in this case study. The use of process mining techniques that can automatically detect loops would facilitate the identification of problems which do not comply with Scrum. Even though the Scrum methodology is not fully supported by Jira Software, we are working on introducing Scrum meetings and roles as a Jira Software add-on which, in the future, will allow this part of the process to be studied as well.

References

[1] S. Overhage, S. Schlauderer, D. Birkmeier, and J. Miller, "What makes IT personnel adopt Scrum? A framework of drivers and inhibitors to developer acceptance," in *44th Hawaii Intl. Conf. on System Sciences (HICSS)*, 2011, pp. 1–10.

[2] A. Marchenko and P. Abrahamsson, "Scrum in a multiproject environment: An ethnographically-inspired case study on the adoption challenges," in *Agile 2008 Conference*. IEEE, 2008, pp. 15–26.

[3] K. Schwaber, *Agile Project Management with Scrum*. Microsoft Press, 2004.

[4] W. M. P. van der Aalst, M. Weske, and D. Grünbauer, "Case handling: a new paradigm for business process support," *Data & Knowledge Engineering*, vol. 53, no. 2, pp. 129–162, 2005.

[5] W. M. P. van der Aalst, *Process mining: Data Science in Action*. Springer, 2016.

[6] R. S. Mans, M. H. Schonenberg, M. Song, W. M. van der Aalst, and P. J. M. Bakker, "Application of process mining in healthcare – a case study in a Dutch hospital," in *Biomedical Engineering Systems and Technologies*, ser. CCIS, vol. 25. Springer, 2009, pp. 425–438.

[7] D. R. Ferreira and M. Mira da Silva, "Using process mining for ITIL assessment: a case study with incident management,," in *Proceedings of the 13th Annual UKAIS Conference*, 2008.

[8] B. Vázquez-Barreiros, D. Chapela, M. Mucientes, M. Lama, and D. Berea, "Process mining in IT service management: A case study," in *International Workshop on Algorithms & Theories for the Analysis of Event Data (ATAED)*, ser. CEUR Workshop Proceedings, vol. 1592, 2016, pp. 16–30.

[9] A. M. Lemos, C. C. Sabino, R. M. F. Lima, and C. A. L. Oliveira, "Using process mining in software development process management: A case study," in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2011, pp. 1181–1186.

[10] M. L. van Eck, X. Lu, S. J. J. Leemans, and W. M. P. van der Aalst, "PM^2: A process mining project methodology," in *Advanced Information Systems Engineering*, ser. LNCS, vol. 9097, Springer. Springer, 2015, pp. 297–313.

[11] M. Jovanović, A.-L. Mesquida, N. Radaković, and A. Mas, "Agile retrospective games for different team development phases," *Journal of Universal Computer Science*, vol. 22, no. 12, pp. 1489–1508, 2016.

[12] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, "Manifesto for agile software development," 2001.

[13] VersionOne, "10th annual state of agile report," Tech. Rep., 2016.

[14] M. Dumas, W. M. P. van der Aalst, and A. H. ter Hofstede, *Process-Aware Information Systems: Bridging People and Software Through Process Technology*. Wiley, 2005.

[15] W. van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: discovering process models from event logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1128–1142, 2004.

[16] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering block-structured process models from event logs containing infrequent behaviour," in *Business Process Management Workshops*, ser. LNBIP, vol. 171. Springer, 2014, pp. 66–78.

[17] A. J. M. M. Weijters, W. M. P. van der Aalst, and A. K. Alves de Medeiros, "Process mining with the heuristics miner-algorithm," Eindhoven University of Technology, BETA Working Paper Series WP 166, 2006.

[18] C. W. Günther and W. M. P. van der Aalst, "Fuzzy mining – adaptive process simplification based on multi-perspective metrics," in *Business Process Management*, ser. LNCS, vol. 4714. Springer, 2007, pp. 328–343.

[19] W. M. P. van der Aalst and M. Song, "Mining social networks: Uncovering interaction patterns in business processes," in *Business Process Management*, ser. LNCS, vol. 3080. Springer, 2004, pp. 244–260.

[20] W. M. P. van der Aalst, B. F. van Dongen, C. W. Günther, A. Rozinat, E. Verbeek, and T. Weijters, "ProM: The process mining toolkit," in *BPM Demo Track*, ser. CEUR Workshop Proceedings, vol. 489, 2009.

[21] C. W. Günther and A. Rozinat, "Disco: Discover your processes," in *BPM Demo Track*, ser. CEUR Workshop Proceedings, vol. 940, 2012, pp. 40–44.

[22] W. Poncin, A. Serebrenik, and M. van den Brand, "Process mining software repositories," in *15th European Conference on Software Maintenance and Reengineering (CSMR)*. IEEE, 2011, pp. 5–14.

[23] S. Bala, C. Cabanillas, J. Mendling, A. Rogge-Solti, and A. Polleres, "Mining project-oriented business processes," in *International Conference on Business Process Management*. Springer, 2015, pp. 425–440.

[24] V. Rubin, C. W. Günther, W. M. P. van der Aalst, E. Kindler, B. F. van Dongen, and W. Schäfer, "Process mining framework for software processes," in *Software Process Dynamics and Agility*, ser. LNCS, vol. 4470. Springer, 2007, pp. 169–181.

[25] V. Rubin, I. Lomazova, and W. M. P. van der Aalst, "Agile development with software process mining," in *International Conference on Software and System Process (ICSSP)*. ACM, 2014, pp. 70–74.

[26] M. Bozkaya, J. Gabriels, and J. M. v. d. Werf, "Process diagnostics: A method based on process mining," in *International Conference on Information, Process, and Knowledge Management*, Feb 2009, pp. 22–27.

[27] A. Syamsiyah, A. Bolt, L. Cheng, B. F. A. Hompes, R. P. Jagadeesh Chandra Bose, B. F. van Dongen, and W. M. P. van der Aalst, *Business Process Comparison: A Methodology and Case Study*. Springer, 2017, pp. 253–267.

[28] C. W. Günther and W. M. P. van der Aalst, "Process mining in case handling systems," *Proceedings of the Multikonferenz Wirtschaftsinformatik*, 2006.

[29] W. M. P. van der Aalst and P. J. S. Berens, "Beyond workflow management: Product-driven case handling," in *International ACM SIGGROUP Conference on Supporting Group Work*, 2001, pp. 42–51.

[30] M. Song and W. M. P. van der Aalst, "Supporting process mining by showing events at a glance," in *17th Annual Workshop on Information Technologies and Systems (WITS)*, 2007, pp. 139–145.