Using Process Mining to Leverage the Development of a Family of Mobile Applications

Lyudmila Rezunik, Alisa Perevoznikova, Daria Eremina, Alexey Mitsyuk
HSE University, Faculty of Computer Science
Moscow, Russia
lrezunic@gmail.com, alice.castiel1@gmail.com, dveremina@edu.hse.ru, amitsyuk@hse.ru

***Abstract*—Enterprises often provide their services via a family of applications based on various platforms. Applications in such a family can behave differently. Their development processes can differ as well. Moreover, modern development processes are often complex and sometimes vague. This can lead to bugs, defects, and unwanted discrepancies in applications. In this paper, we show that process mining can be applied to leverage the development in such a case. Real-life models can be discovered and investigated by the developer teams in order to reveal differences in application behaviour, find bugs, and highlight inefficiencies. We consider datasets with event data of two types. Firstly, we analyse event logs of Android and iOS applications of the same product family. Secondly, we consider event data from working repositories of these applications. We show how by analysing such datasets, the real-life development process can be discovered. Besides, application event logs can help to find more and less severe bugs and unwanted behaviour.**

***Index Terms*—software process, software development, process mining, mobile application, software product family**

I. Introduction

It is a common practise for business enterprises to provide their services via different user applications. We can, for example, use a web-application at our desktop and a mobile application when outdoors. Moreover, users have different mobile devices based on various technologies. All this leads to a family of applications that is developed and maintained by an enterprise to provide its services directly to potential users on their familiar platforms. Companies can develop members of such a software product family separately, one-by-one, or maintain a common software development environment. Combined with modern agile development approaches, all this leads to high variability, complexity, and sometimes vagueness of development process. This, in turn, can lead to bugs and defects in software.

To our fortune, software applications on all platforms generate a large number of data records in the process of their functioning. Different types of data are present: user activity logs, error and system logs, debug information, communication logs, and other. We can use these records to discover actual development process, find its inefficiencies and drawbacks. Moreover, an investigation of application's event logs can shed the light on its structure and behaviour (see Section VIII). Process mining [1], [2] is a particular field providing us with tools which help to extract valuable information and insights out of raw event data.

In this paper, we consider two datasets (see Section IV) for a concrete family of mobile applications (see Section II for a system description). The first of these datasets contains event data from the repositories with source code. The second one has been obtained by recording logs of how users interact with the applications.

The main goal of this paper is twofold: (1) to show how can we reveal the *real* development process of a family of mobile applications using event data, (2) to provide the reader with the approach to find drawbacks and errors in both applications and their development process.

Our case study is conducted in accordance with the PM^2 methodology [3] for process mining projects. By analysing applications log data, companies aim to improve their business processes [4]. We show that this methodology can be successfully applied within the domain of software engineering with valuable outcomes.

II. System

The analysis was conducted for a family of mobile applications — HSE App X[1], which includes iOS and Android applications with the same functionality.

HSE App X as a whole is a client-server application used by students and staff of HSE University[2] to interact with the university's systems. As it was mentioned, there are two client applications — for iOS and Android.

Taking as an example the iOS application, it can be seen how the client is built (the access to the project repository was granted by the developers).

The client contains several modules: a module for authorization functionality, a module for Apple Watch application, widgets, and the main module, which represents the iOS application (see Fig. 1).

The main module is organised into groups of files (packages): Core, UI, Assets, Helpers. A summary of each package is provided below.

Core — includes sub-packages which implement the logic for authorisation, system events, API calls. This package also contains the main entry point of the application.

This work is an output of a research project implemented as part of the Basic Research Program at the National Research University Higher School of Economics (HSE University).

[1]HSE App X Official Webpage: https://www.hse.ru/web/mobile/#
[2]HSE University Official Webpage: https://www.hse.ru

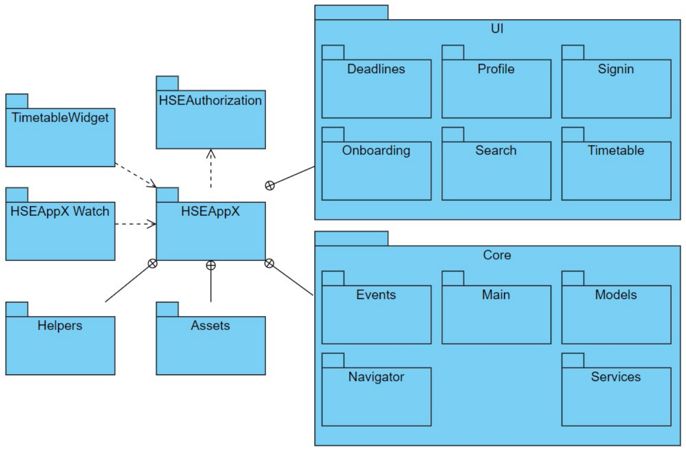


Fig. 1. Package diagram of the iOS client

UI — contains separate packages for each of the applications' screens.

Assets — package with various media files (images, sounds). Moreover, it stores all the fonts and colours used in the application along with localisation.

Helpers — holds helper classes, extensions for existing classes, mocks.

An example of typical user scenario is viewing the timetable. What happens on application start:

1) All the classes that need to connect to the API are initialised.
2) The application sends requests to the API to get:
   a) the catalogues (e. g. all the HSE buildings) from the server;
   b) the user's notifications;
   c) information about the user;
   d) list of features available to a certain user.

Along with sending requests, the first screen of application loads (in parallel). As soon as the screen controller object is created, the authorisation token is verified and the request to get user's schedule is sent. After receiving a response, the application shows all the data on screen and saves it in cache.

Both Android and iOS application are logging results of API calls, system events and user activity. Thus, the logs can be effectively used for analysis.

III. Research Tasks and Questions

In this paper, our goal is to perform the following tasks and to answer related research questions:

1) to build a process model for various user scenarios and determine if there are any "abnormal" (deviating from the norm) events. If there are any, try to explain them;
2) to analyse the process model for iOS and Android applications and find differences. If there are any, find out if they affect the performance and operation of the system, whether they need to be resolved;
3) to check if there are inconsistencies between the request body and the server response;
4) to identify the sequences within the process model that lead to error with the greatest frequency;
5) to analyse the data from the project repository and evaluate the team's work style;
6) to track which parts of the code (modules, files) have not been refactored/redesigned for the longest time and that are worth paying attention to.

IV. Data

A. Data

In order to answer the research questions, two main sources of data were taken into consideration: logs of mobile applications and the project's repository.

Mobile applications log the user's actions, making it possible to extract files with a detailed history. They contain all the necessary information about the time and content of each request sent to the server and system logs as well. The developers are provided with different options to interact with debug mode of the application (it is hidden from regular users), which is helpful in obtaining suitable data for analysis. For these purposes it is necessary to clear previous session logs, so it becomes possible to focus on the particular user's interactions.

Log files collected for particular use cases of the application were parsed by found patterns and then aggregated using common Python libraries. As a result, the data contained structured information about requests' URL, body and response with corresponding timestamps.

The repository stores all the project members' activity: commits, merge requests, issues, etc. This data can be effectively used to analyse the processes within the team. Considering the research questions, it was identified that commits hold the most information. Thus, all the commits stored in the GitLab repository were extracted into a single .csv file. For data retrieval, a few scripts were written using the Java library GitLab4J [3] to make the process of working with GitLab's API more straightforward.

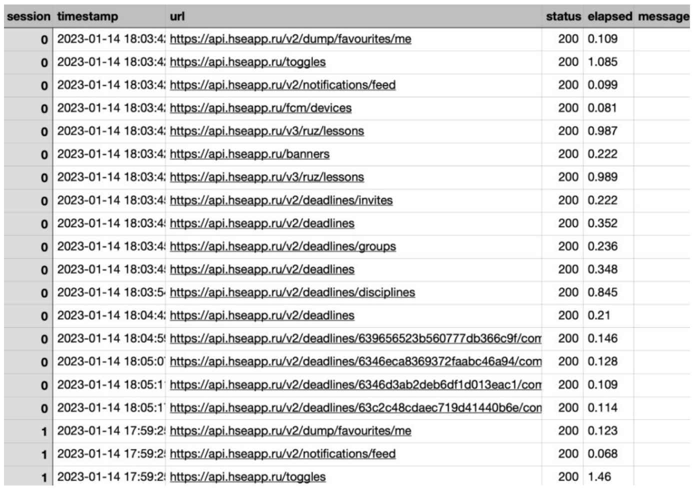
The resulting data file contains all the needed information about the commits: id, author's name, title of commit, all the changes (file paths), and timestamp, which corresponds to the exact date and time the commit was made locally (before push)[4] .

B. Data Preprocessing

The previous section described the process of collecting the data from the project's repository, resulting in only the necessary data for analysis remaining, including commit id, date and time of commit creation, author's name, and all the names of changed files.

[3]GitLab4J API: https://github.com/gitlab4j/gitlab4j-api

[4]The data collected from the project's repository: https://github.com/LucyRez/PAMiSE-data/tree/master/gitlab-data



session	timestamp	url	status	elapsed	message
0	2023-01-14 18:03:4	https://api.hseapp.ru/v2/dump/favourites/me	200	0.109	
0	2023-01-14 18:03:4	https://api.hseapp.ru/toggles	200	1.085	
0	2023-01-14 18:03:4	https://api.hseapp.ru/v2/notifications/feed	200	0.099	
0	2023-01-14 18:03:4	https://api.hseapp.ru/fcm/devices	200	0.081	
0	2023-01-14 18:03:4	https://api.hseapp.ru/v3/ruz/lessons	200	0.987	
0	2023-01-14 18:03:4	https://api.hseapp.ru/banners	200	0.222	
0	2023-01-14 18:03:4	https://api.hseapp.ru/v3/ruz/lessons	200	0.989	
0	2023-01-14 18:03:4	https://api.hseapp.ru/v2/deadlines/invites	200	0.222	
0	2023-01-14 18:03:4	https://api.hseapp.ru/v2/deadlines	200	0.352	
0	2023-01-14 18:03:4	https://api.hseapp.ru/v2/deadlines/groups	200	0.236	
0	2023-01-14 18:03:4	https://api.hseapp.ru/v2/deadlines	200	0.348	
0	2023-01-14 18:03:5	https://api.hseapp.ru/v2/deadlines/disciplines	200	0.845	
0	2023-01-14 18:04:4	https://api.hseapp.ru/v2/deadlines	200	0.21	
0	2023-01-14 18:04:5	https://api.hseapp.ru/v2/deadlines/639656523b560777db366c9f/com	200	0.146	
0	2023-01-14 18:05:0	https://api.hseapp.ru/v2/deadlines/6346eca8369372faabc46a94/com	200	0.128	
0	2023-01-14 18:05:1	https://api.hseapp.ru/v2/deadlines/6346d3ab2deb6df1d013eac1/com	200	0.109	
0	2023-01-14 18:05:1	https://api.hseapp.ru/v2/deadlines/63c2c48cdaec719d41440b6e/com	200	0.114	
1	2023-01-14 17:59:2	https://api.hseapp.ru/v2/dump/favourites/me	200	0.123	
1	2023-01-14 17:59:2	https://api.hseapp.ru/v2/notifications/feed	200	0.068	
1	2023-01-14 17:59:2	https://api.hseapp.ru/toggles	200	1.46	

Fig. 2. Example of a dataset collected from the Android application

In addition, it was decided that commits should be also grouped in some way, because the number of individual commits was too large to be efficiently processed by some process mining tools. Knowing that the project was managed using Agile methodology, sprint number was added to the each of the commits (sprint is a 2 week long time interval). Thus, each of the commits was assigned a sprint it was created in. This also improved the dataset, because not only time of creation of a single commit could be used for analysis, but also a group of commits.

Besides, the logs of iOS and Android applications were collected separately[5], after which they were converted into a format suitable for processing. After processing the data, it acquired the following form: session id, date and time of the request, request URL (without parameters), response status, duration of the request, message (see Fig. 2).

At the same time, both the request and the response are logged in the Android application, so it was necessary to remove duplicates from the file. Also, in the Android application, in case of incorrect requests, incorrect dates appeared that had to be processed. To determine the case id, the session number was added (a separate application launch), and the date and time format was also changed to fit the mining algorithms provided by mining tools. Additional parameters were removed from the URLs that would interfere with building a graph due to the presence of too specific information.

V. Development Process Analysis

A. Analysis

The dataset from the previous section, which was formed using repository data, was applied to interpret the processes among the developers. Analysis of the data was conducted using ProM [5], a program specifically developed for process analysis. It provides a big tool set, which was used throughout all research.

[5]The examples of logs: https://github.com/LucyRez/PAMiSE-data/tree/master/logs-data

Firstly, the team workflow was analysed by generating dotted chart diagrams with different display options. For example, by putting the value of the commit creation time (since the start of the week) on the X axis and commit's id on the Y axis, a chart representation of developer's working schedule was built (see Fig. 3).

Moreover, the time of creation of all commits was analysed. The constructed dotted chart gave representation of developers' individual working hours.

This method of analysis was also used to discover project members' involvement into development process. It can be seen through the dependency between changes in project modules and the author of the commit. The diagram (see Fig. 4) illustrates the division of duties among the developers, and shows which pieces of code can only be changed by a particular person.

In addition to the dotted chart diagrams, which resemble statistical research methods, some models were synthesised by applying the inductive mining algorithm (using the Inductive Miner [6] inside ProM). These models were used to see the transfer of work (if it exists) between the developers. Instead of inspecting individual commits, groups of commits were taken into consideration (each group was represented by sprint number).

The inductive miner algorithm requires a dataset with 2 columns selected. The first column represents case id, which identifies a single trace of the process (in this case it is one sprint — 2 weeks). The second selected value type — activity, which basically is an event in the trace (name of commit's author was used).

Thus, the resulting model (see Fig. 5) shows the generalised behaviour of the team members during one sprint. Conclusions on this model will be given in the next section of the document.

Another approach for viewing data, which was used in the research — generating a skeleton of the event journal. This model can give a clear understanding of the dependencies between the execution of events in one trace of the process. If the trace is represented by a single commit and the events in the trace are file changes, the resulting model can depict which files do not ever change together in a commit or vice versa.

B. Main Findings

After analysing the generated dotted chart diagrams, it was noticed that the team does not have a strict working schedule, commits in the repository are made every day of the week. However, there is a tendency among the developers to leave the weekends to themselves, that is why there is less activity on Sunday and Saturday.

Moreover, the working hours are not established for any of the team members. Distribution of the commits covers almost all the area of a one-day timeline. The developers can make changes to the project even in the early morning, although the peak activity is in the daytime.

The dotted chart diagram made it possible to assess in more detail the degree of familiarity of developers with individual

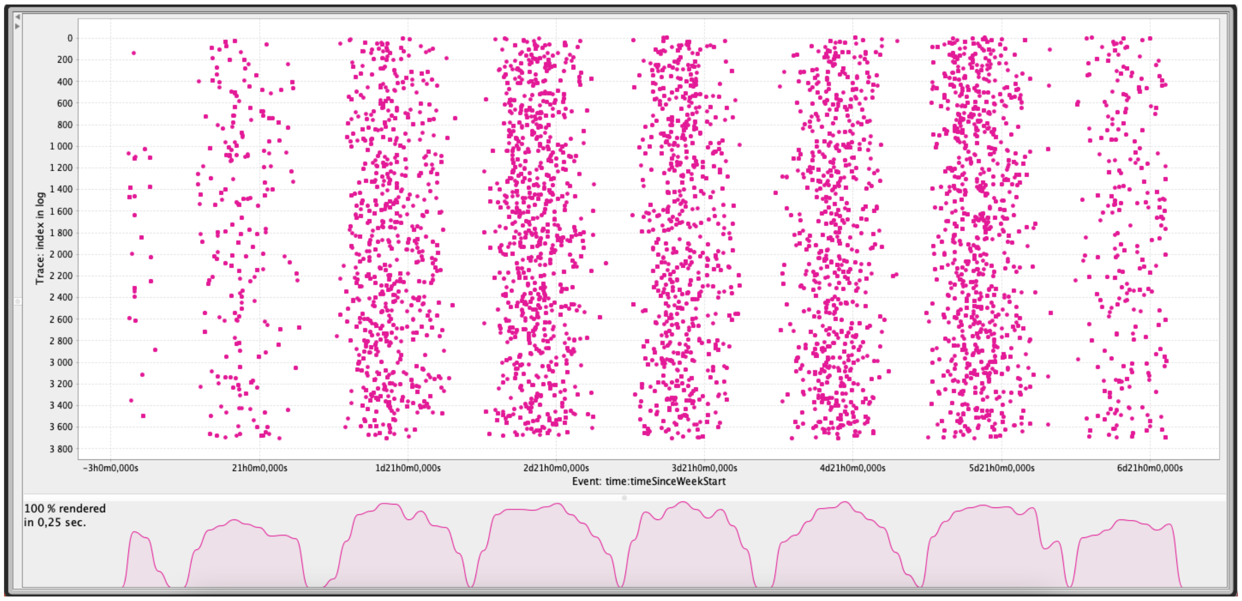


Fig. 3. Dotted chart that shows developers' working schedule

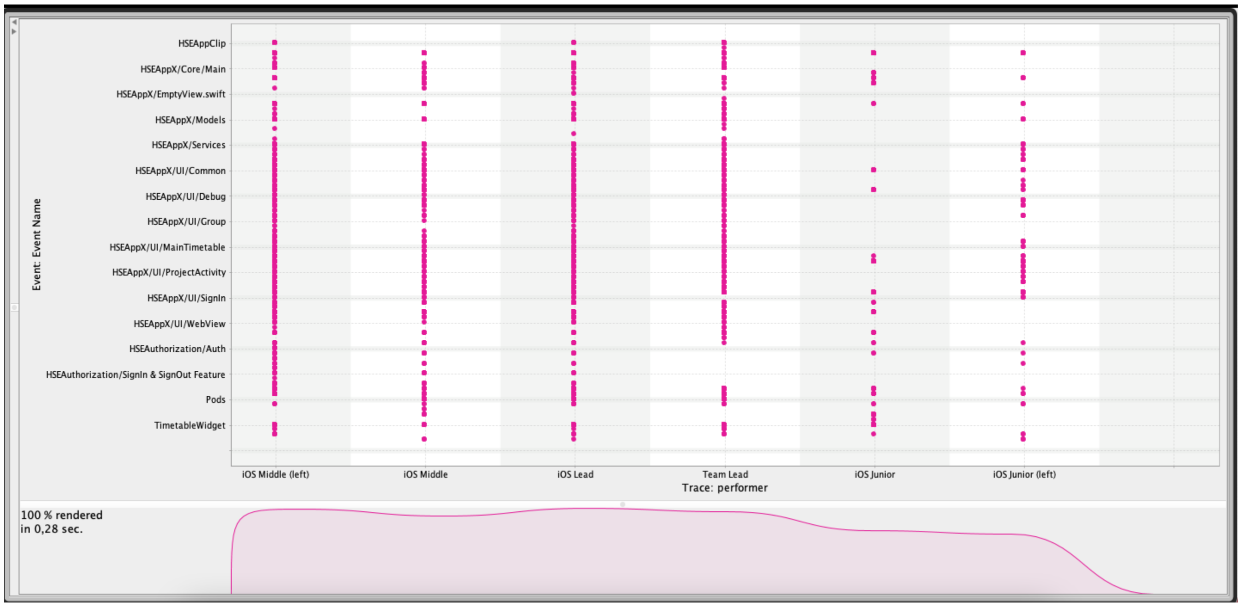


Fig. 4. Dotted chart shows the commits submitted by developers for certain modules

modules of the project. For example, it was noticed, that the person which was no longer on the team was initially working on the authorisation module, and the other developers' tasks only included small changes and refactoring of that code.

By applying the inductive miner algorithm (described in the previous section) was produced a model, which depicts the order in which the developers make commits in a sprint. As it can be seen there is no dependency between the developers. The team members work in parallel, there are situations when a person does not make a single commit per sprint, but there is no certain sequence of work transfer between developers. This also shows that the tasks are mostly independent of each other.

It was also discovered that some developers never co-existed in the team: iOS Middle, both of the iOS Juniors and HSE Apps, which is another account run by project manager.

The Log Skeleton Visualiser [7] was applied to discover relations between packages in a commit (if there are such relations). Thus, the case id for the mining algorithm was represented by the id of a single commit and the activity was represented by the changed file.

It was noticed that some changes never co-occur in a commit. Developers try to keep their commit history clean, thus, it is not favourable to make changes in completely

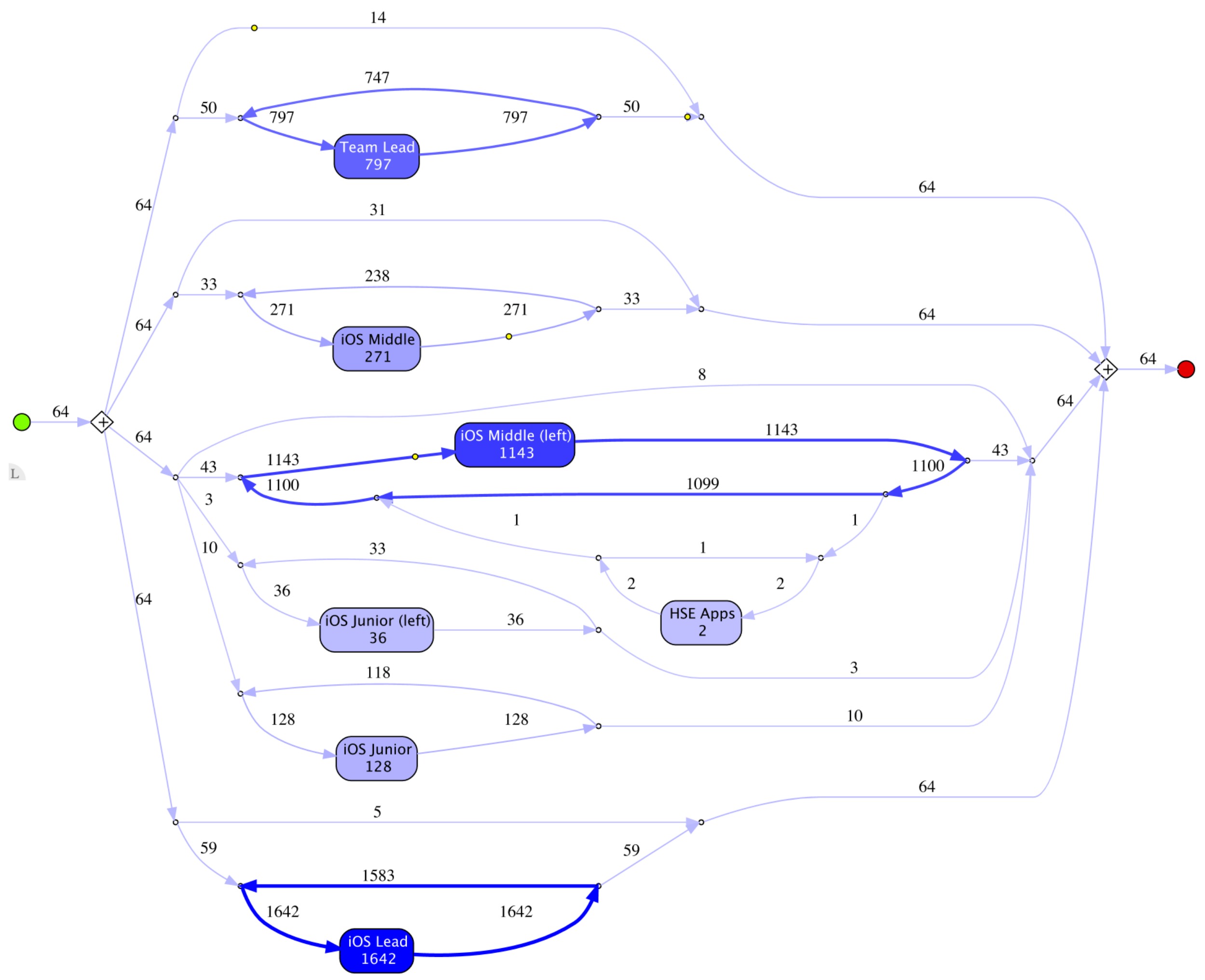


Fig. 5. Model depicting activity of all developers during one sprint

separate modules at once.

Taking for example Apple Watch module and Widget module, it is true that project members try to change files in those modules separately (see Fig. 6).

While looking at the statistics overview for all the commits, it was noticed that the number of changed files in one commit has a strong variation: from 1 changed file up to 693 files.

It is strongly advised to not have too many file changes in a single commit, because the code needs to go through a more in-depth code review. The developer in that case cannot keep track of all the changes, with a high probability such code may be bugged.

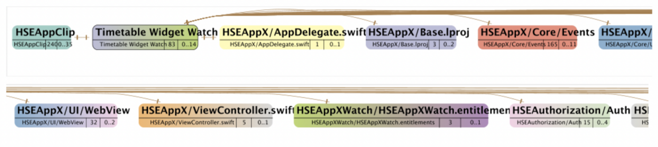


Fig. 6. Fragment of the log skeleton model depicting relations between file changes

VI. Application Behavior Analysis

A. Analysis

For each dataset one of the application usage scenarios from the following list was reproduced:

1) *Viewing a personal timetable*
The user gets to the main screen of the application, looks at his timetable, and then opens the page of a certain lecture (or another type of activity) to view information about it.

2) *Search for a person's timetable*
The user goes to the search section from the main screen, searches for the person, looks through his timetable.

3) *Search for free classrooms*
The user goes to the services section from the main screen, configures the parameters for searching for classroom (building, date, time), after which he receives a list of available classrooms and views the schedule of a specific one.

4) *Viewing the grade book*
   The user goes to the profile screen from the main screen, where he selects his grade book, a year of study, a discipline for which statistics should be viewed. After that, he returns to the screen with the grade, and clicks on the cell with his rating to view the full rating list.
5) *Deadlines*
   The user goes to the deadlines section from the main screen, creates a new personal deadline, sets the necessary parameters (discipline, title, description, participants, time), saves it and marks it as completed. Similar sequence of actions should be performed in case when the user creates a new group to add a common deadline.

To compare the conformance for different scenarios, Log Skeleton models were built. Using the example of one scenario, it can be seen that the models are generally similar to each other, they send requests to the same URLs, and in full use case scenario go all the way to the certain disciplines (in case of the first scenario).

At the same time, separate models were built that take into account the responses and errors returned by the requests (see Fig. 7).

Such anomalies were found in almost all scenarios, which made it possible to analyse the causes of such errors. For example, the Android application does send requests that are cancelled afterwards to avoid "race condition" in the application, which confirmed the detected problem.

In addition, Petri nets were built for each process by applying Inductive Miner. Using the example of the "Timetable View" scenario, there is a noticeable discrepancy in its execution of the passage from the main screen to the necessary target (the screen of a specific discipline in the timetable). However, it is clear that applications send similar requests and reach similar endpoints (see Fig. 8).

B. Main Findings

A research study of mobile application logs was conducted to compare the operations of iOS and Android mobile applications, with a focus on identifying differences and unexpected behaviour patterns. The dataset of mobile application logs was analysed with the aim of uncovering insights that could help the business and the developers optimise their mobile apps for each platform. In this section, the main findings of this study are presented, highlighting the key differences and similarities between iOS and Android mobile applications.

First of all, in the study was used the Log Visualiser and summaries for each use-case to gain insights into the types of server requests being made on each platform. The Log Visualiser provided a graphical representation of the logs, while the summaries allowed for a more detailed analysis of the data. The analysis of the logs revealed that there were similar server requests being made on both platforms, with matching URLs. However, the frequency of these requests varied depending on the platform, due to the involvement of different screens. Despite these differences, the behaviour of the mobile applications on both iOS and Android platforms was found to be quite similar. Overall, this part of research contributes to a better understanding of high-level application behaviour, while next mining approaches helped to find out more detailed information.

After summaries analysis, several models were built to discover any differences in performed activities and their drawbacks. The Petri Nets were used to visualise the traces of the mobile applications. The use of such models in this study was beneficial as it allowed to identify potential bottlenecks, because requests were the same at first sight and had no significant findings. However, there were some unexpected steps in traces. This led to usage of the Log Skeleton Visualiser in ProM, which provided a detailed graph of requests and their corresponding responses, allowing for a more comprehensive analysis of the mobile application logs. One of the key findings of this research was that the Android application logs graphs almost always contained an unexpected node with the error message 'java.io.Exception: Canceled'. This finding prompted to take a closer look at the processes involved in the Android application logs. Upon closer examination, it was discovered that there were confirmed issues with sending requests and cancelling them due to race conditions, which were likely contributing to the found type of error in the Android application logs graphs. Moreover, a repeating error in some log traces of the iOS application was found. It was caused by incorrect parsing of the data that was coming from the server.

This research highlighted the need to address the issues in the Android application to improve its performance and reliability. Such analysis can help developers to optimise their software and prevent similar issues from arising in the future.

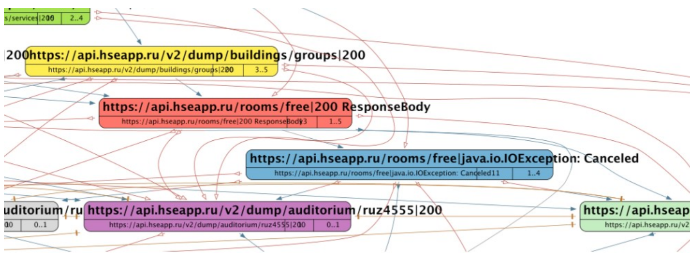

s/services|200 2..4
200https://api.hseapp.ru/v2/dump/buildings/groups|200
https://api.hseapp.ru/v2/dump/buildings/groups|200 3..5
https://api.hseapp.ru/rooms/free|200 ResponseBody
https://api.hseapp.ru/rooms/free|200 ResponseBody3 1..5
https://api.hseapp.ru/rooms/free|java.io.IOException: Canceled
https://api.hseapp.ru/rooms/free|java.io.IOException: Canceled11 1..4
uditorium/ru
00 0..1
https://api.hseapp.ru/v2/dump/auditorium/ruz4555|200
https://api.hseapp.ru/v2/dump/auditorium/ruz4555|200 0..1
https://api.hseap
https://api.hseapp.ru/v2

Fig. 7. Fragment of the log skeleton visualisation of the schedule viewing scenario in an iOS application with detected errors

VII. Discussion

During the research of application processes, Android and iOS logs were collected and prepared for analysis, as well as a representation of individual scenarios in the form of models obtained from them. To do this, it was necessary to clear the logs of useless information, parse the needed parts, bring them to a tabular form and select columns for analysis. This made it possible to track the behaviour on various platforms, to identify patterns and anomalies in the requests, which should lead to the same result.

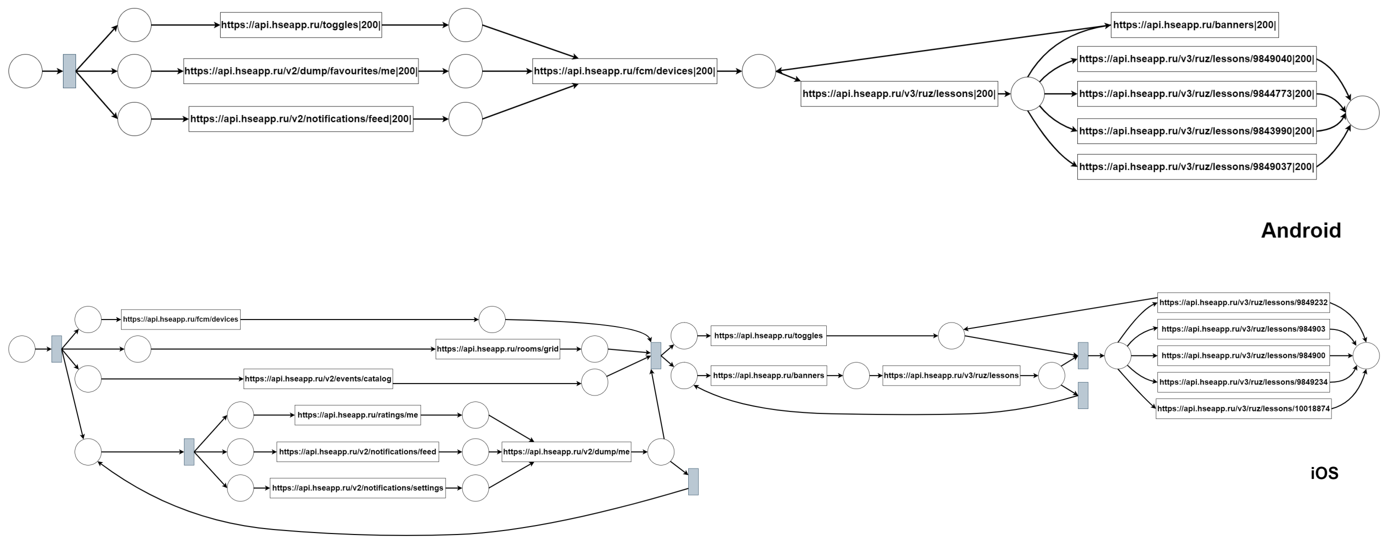

https://api.hseapp.ru/toggles|200|
https://api.hseapp.ru/v2/dump/favourites/me|200|
https://api.hseapp.ru/v2/notifications/feed|200|
https://api.hseapp.ru/fcm/devices|200|
https://api.hseapp.ru/v3/ruz/lessons|200|
https://api.hseapp.ru/banners|200|
https://api.hseapp.ru/v3/ruz/lessons/9849040|200|
https://api.hseapp.ru/v3/ruz/lessons/9844773|200|
https://api.hseapp.ru/v3/ruz/lessons/9843990|200|
https://api.hseapp.ru/v3/ruz/lessons/9849037|200|
Android
https://api.hseapp.ru/fcm/devices
https://api.hseapp.ru/rooms/grid
https://api.hseapp.ru/v2/events/catalog
https://api.hseapp.ru/ratings/me
https://api.hseapp.ru/v2/notifications/feed
https://api.hseapp.ru/v2/notifications/settings
https://api.hseapp.ru/v2/dump/me
https://api.hseapp.ru/toggles
https://api.hseapp.ru/banners
https://api.hseapp.ru/v3/ruz/lessons
https://api.hseapp.ru/v3/ruz/lessons/9849232
https://api.hseapp.ru/v3/ruz/lessons/984903
https://api.hseapp.ru/v3/ruz/lessons/984900
https://api.hseapp.ru/v3/ruz/lessons/9849234
https://api.hseapp.ru/v3/ruz/lessons/10018874
iOS

Fig. 8. Petri nets depicting the process of viewing the schedule using Android and iOS applications

A. Answers to Research Questions

During the analysis of diagrams, models and the general report in relation to a set of different scenarios, it was found out that there are no critical discrepancies regarding the requests sent, since the requests URLs, their number, as well as the overall behaviour on the traces coincided. Thus, it can be assumed that there are no abnormal events and discrepancies in the general case. This answers the first research question.

To answer the second research question a more detailed analysis of the logs collected in one scenario was conducted. It allowed to identify erroneous events that are often found only in the Android application. On Log Skeleton models, it was noticeable that an event with a request cancellation error appears in the response. There were no dependencies on specific scenarios, but it was clear that such a response was returned as a result of sending a duplicate request that had already left the application. Serialisation errors were noticed for the iOS application, which did not lead to further deviations, but occurred in several traces.

It was also found that the sequences which are more prone to errors are those which include search requests (for Android). For iOS such obvious sequences were not found. There also weren't noticed any inconsistencies between the response data and application requests. Thus, the research questions 4 and 3 were also answered.

As a result of analysing the data from the repository, it became possible to understand the team's work style (the daily routine, working days of the week, methodology of development, quality of commits — size, which parts of the code were affected).

As it turned out, the developers' work schedule is not strictly set, each team member independently determines when it is more convenient for him to work. It can be noticed that developers commit to the repository less often on Saturday and Sunday. That is, programmers, even if they are not limited in the choice of working hours, decide not to work on weekends. In the same way, it is noticeable that commits are made at completely different times.

Project participants work on tasks in parallel with each other and everyone's familiarity with the project is approximately on the same level. Everyone has worked with almost every file in the repository at some point in time. The only exception is the junior developer, which is logical, given his level.

It was also possible to notice that commits are carried out in all modules of the project so far. It can be assumed that the tasks relate to different parts of the project and they can be issued to different developers. This approach to working on software products is also popular in large companies — when new functionality is gradually added to the product over time and when needed. That suggests any of the Agile methodologies is used in the development process (or no methodology is used at all, and programmers work the way they are used to). This concludes the answer of the fifth question of this research.

It was also possible to identify the parts of the project in which commits have been carried out the least recently (the last research question). The informativeness of such a list is a little doubtful, since there are several auxiliary files (for example, SberPaySDK[6]). There are also many references to the authorisation module and to the main module of the project. With the help of additional tools, it was possible to identify several files where commits have not been performed for a long time — these are files related to network interaction and some basic files. However, this information is not accurate enough, since the analysis included large commits affecting almost all project files.

[6]Sber API Registry: https://api.developer.sber.ru/product/SberbankID/doc/v1/iOSsdk

B. Open Questions and Problems

Among the open questions regarding the various versions of the application, is an in-depth comparison of the body of the response to the request with the current status. To do this, it is necessary to improve parsing, explore examples of responses and such cases in logs, on the basis of which it will be possible to build new models. The detection of such discrepancies will improve the responses from the service, which will affect their processing in applications. It would be better to test the application work on a large number of scenarios and with the participation of real users.

VIII. Related Work

This section is a brief review of the field of mining software data. Process mining considers software as a research object for more than fifteen years [8], [9], and a lot of contributions have been made in this field. Most of them can be grouped into two major classes depending on what processes are considered:

- software behaviour,
- software (development, maintenance etc.) process.

Let us consider papers of both these classes. In this paper, we begin with the analysis of a development process. Therefore, in this section, we will follow the same approach.

The problem of team work assessment is crucial in the domain [10]. Indeed, process mining can be used to evaluate process performance. Software development process can be evaluated more or less in the same way as other business processes. To achieve more detailed results, mining of event logs can be combined with more conventional approaches like surveys and expert evaluation [11].

Lightweight development approaches are a target for mining due to their agility and non-linear nature [12]. Marques et at. [13] evaluates programmer's activities within agile development teams. The aim is to find what Scrum disciplines and practises can be revealed and checked using event logs from a case-handling system. Interestingly, some of practises can be discovered using very basic techniques.

Mining event logs of individual programmers is another emerging sub-domain [14], [15]. Process mining techniques can be used, for example, to assist and fine-tune learning process of novice developers [16]. Supplementary data from repositories of student projects can be investigated as well [17]. Such an analysis gives many ideas of how to improve and evolve existing programming courses in a more interactive learning facilities. Security training of programmers and users can benefit of user log mining no less than other sub-domains [18], [19].

Let us now consider applications of process mining to software behaviour analysis.

Process analysis methods can be applied to the very low-level technical problems. For example, Wakup and Desel [20] considered how logs of an application with active TCP/IP information interchange can be analysed using process mining. Authors constructed models of client-server communication with both sides clearly separable.

Leemans et al. proposed [21] and later developed [22] a methodology to analyse event logs of software systems. The goal is to reverse engineer a (possibly, legacy) system to reveal and grasp its behavioural characteristics. The methodology combines a general approach to software process analysis (similar to PM^2 [3]) with very technical tools [23]. Leemans et al. even constructed a ProM plug-in — *Statechart Workbench* [24] — that can support users of this approach.

In 2015, Shershakov and Rubin published a paper [25] on how to analyse real-life software executions and what can be discovered from event logs of such executions. Later, the team of collaborators of Sergey Shershakov developed several techniques to synthesise UML activity diagrams and other types of visual model for service-oriented and component-based systems [26], [27].

The same component-based systems have been analysed by Liu with his co-authors [28]–[30] In a series of papers, they worked on a problem of identifying communicating components and interfaces through which components communicate. Separate services in an enterprise-scale service-oriented system can be discovered based on event log analysis as well [31]

Process analysis approaches can be valuable in more specific sub-domains of software engineering. Software reliability can be assessed based on process mining techniques. A lot of approaches reviewed by Macák et al. [32] Interestingly, event the evolution of highly-distributed systems like blockchain applications can be successfully revealed using process analysis approaches [33].

Mobile applications can also be the object for process analysis. Process mining is a popular research topic in various fields related to user applications, such as mobile games [34] and others. Usually, the goal of the research projects is to solve some common problem and find various (anti-)patterns in user behaviour. Sometimes, researchers are trying to answer business-related questions for particular mobile applications [35]. Log analysis for a specific platform was also mentioned in literature [36], but without the goal to compare behaviour of several applications for different platforms.

From this concise literature review, we conclude that as software generates a lot of data, this data can be analysed with valuable outcomes for developers, designers, and users. The field of software process analysis develops actively. Methods applied in our paper are in line with the state-of-the-art approaches in process mining. The object of our investigation — a family of mobile applications — is new for the field.

IX. Conclusion

In this paper, we presented the results of a case study. We applied process mining to analyse datasets containing event data recorded within the development process of mobile applications. Usually, mobile applications are developed and maintained as a family because enterprises want to provide their services on various platforms. In the paper, we show that different applications of the same family can behave in a variety of ways. Besides, their development processes can differ as well. Process mining allows for discovering real process models which can be easily investigated by developer teams in order to reveal unwanted discrepancies, find bugs, and highlight inefficiencies.

Acknowledgements

This work is an output of a research project implemented as part of the Basic Research Program at the National Research University Higher School of Economics (HSE University).

References

[1] W. M. P. van der Aalst, *Process Mining - Data Science in Action, Second Edition*. Springer, 2016.

[2] W. M. P. van der Aalst and J. Carmona, Eds., *Process Mining Handbook*, ser. Lecture Notes in Business Information Processing. Springer, 2022, vol. 448.

[3] M. L. van Eck, X. Lu, S. J. J. Leemans, and W. M. P. van der Aalst, "PM ˆ2 : A process mining project methodology," in *CAiSE*, ser. Lecture Notes in Computer Science, vol. 9097. Springer, 2015, pp. 297–313.

[4] T. W. W. M. van der Aalst and L. Maruster, "Workflow mining: Discovering process models from event logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. v. 16, n. 9, 2004.

[5] E. Verbeek, J. C. A. M. Buijs, B. F. van Dongen, and W. M. P. van der Aalst, "Prom 6: The process mining toolkit," in *BPM (Demos)*, ser. CEUR Workshop Proceedings, vol. 615. CEUR-WS.org, 2010.

[6] S. J. J. Leemans, *Robust Process Mining with Guarantees - Process Discovery, Conformance Checking and Enhancement*, ser. Lecture Notes in Business Information Processing. Springer, 2022, vol. 440.

[7] H. M. W. Verbeek, "The log skeleton visualizer in prom 6.9," *Int. J. Softw. Tools Technol. Transf.*, vol. 24, no. 4, pp. 549–561, 2022.

[8] V. A. 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 *ICSP*, ser. Lecture Notes in Computer Science, vol. 4470. Springer, 2007, pp. 169–181.

[9] V. A. Rubin, A. A. Mitsyuk, I. A. Lomazova, and W. M. P. van der Aalst, "Process mining can be applied to software too!" in *ESEM*. ACM, 2014, pp. 57:1–57:8.

[10] J. Caldeira, F. B. e Abreu, J. P. dos Reis, and J. Cardoso, "Assessing software development teams' efficiency using process mining," in *ICPM*. IEEE, 2019, pp. 65–72.

[11] D. Vavpotic, S. Bala, J. Mendling, and T. Hovelja, "Software process evaluation from user perceptions and log data," *J. Softw. Evol. Process.*, vol. 34, no. 4, 2022.

[12] V. A. Rubin, I. A. Lomazova, and W. M. P. van der Aalst, "Agile development with software process mining," in *ICSSP*. ACM, 2014, pp. 70–74.

[13] R. Marques, M. M. da Silva, and D. R. Ferreira, "Assessing agile software development processes with process mining: A case study," in *CBI (1)*. IEEE Computer Society, 2018, pp. 109–118.

[14] C. Ioannou, A. Burattin, and B. Weber, "Mining developers' workflows from IDE usage," in *CAiSE Workshops*, ser. Lecture Notes in Business Information Processing, vol. 316. Springer, 2018, pp. 167–179.

[15] P. Ardimento, M. L. Bernardi, M. Cimitile, and F. M. Maggi, "Evaluating coding behavior in software development processes: a process mining approach," in *ICSSP*. IEEE / ACM, 2019, pp. 84–93.

[16] P. Ardimento, M. L. Bernardi, M. Cimitile, and G. D. Ruvo, "Learning analytics to improve coding abilities: a fuzzy-based process mining approach," in *FUZZ-IEEE*. IEEE, 2019, pp. 1–7.

[17] M. Macák, D. Kruzelova, S. Chren, and B. Buhnova, "Using process mining for git log analysis of projects in a software development course," *Educ. Inf. Technol.*, vol. 26, no. 5, pp. 5939–5969, 2021.

[18] M. Macák, R. Oslejsek, and B. Buhnova, "Process mining analysis of puzzle-based cybersecurity training," in *ITiCSE (1)*. ACM, 2022, pp. 449–455.

[19] ——, "Applying process discovery to cybersecurity training: An experience report," in *EuroS&P Workshops*. IEEE, 2022, pp. 394–402.

[20] C. Wakup and J. Desel, "Analyzing a tcp/ip-protocol with process mining techniques," in *Business Process Management Workshops*, ser. Lecture Notes in Business Information Processing, vol. 202. Springer, 2014, pp. 353–364.

[21] M. Leemans and W. M. P. van der Aalst, "Process mining in software systems: Discovering real-life business transactions and process models from distributed systems," in *MoDELS*. IEEE Computer Society, 2015, pp. 44–53.

[22] M. Leemans, W. M. P. van der Aalst, M. G. J. van den Brand, R. R. H. Schiffelers, and L. Lensink, "Software process analysis methodology - A methodology based on lessons learned in embracing legacy software," in *ICSME*. IEEE Computer Society, 2018, pp. 665–674.

[23] M. Leemans, W. M. P. van der Aalst, and M. G. J. van den Brand, "Recursion aware modeling and discovery for hierarchical software event log analysis (extended)," *CoRR*, vol. abs/1710.09323, 2017.

[24] ——, "The statechart workbench: Enabling scalable software event log analysis using process mining," in *SANER*. IEEE Computer Society, 2018, pp. 502–506.

[25] S. A. Shershakov and V. A. Rubin, "System runs analysis with process mining," *Modeling and Analysis of Information Systems*, vol. 22, no. 6, pp. 818–833, 2015. [Online]. Available: https://www.mais-journal.ru/jour/article/view/297

[26] K. V. Davydova and S. A. Shershakov, "Mining hybrid uml models from event logs of soa systems," *Proceedings of the Institute for System Programming of the RAS (Proceedings of ISP RAS)*, vol. 29, no. 4, pp. 155–174, 2018. [Online]. Available: https://ispranproceedings.elpub.ru/jour/article/view/319

[27] N. S. Zubkova and S. A. Shersakov, "Method for building uml activity diagrams from event logs," *Proceedings of the Institute for System Programming of the RAS (Proceedings of ISP RAS)*, vol. 31, no. 4, pp. 139–150, 2019. [Online]. Available: https://ispranproceedings.elpub.ru/jour/article/view/1200

[28] C. Liu, B. F. van Dongen, N. Assy, and W. M. P. van der Aalst, "Component behavior discovery from software execution data," in *SSCI*. IEEE, 2016, pp. 1–8.

[29] ——, "Component interface identification and behavioral model discovery from software execution data," in *ICPC*. ACM, 2018, pp. 97–107.

[30] ——, "A general framework to identify software components from execution data," in *ENASE*. SciTePress, 2019, pp. 234–241.

[31] A. A. C. D. Alwis, A. Barros, A. Polyvyanyy, and C. J. Fidge, "Function-splitting heuristics for discovery of microservices in enterprise systems," in *ICSOC*, ser. Lecture Notes in Computer Science, vol. 11236. Springer, 2018, pp. 37–53.

[32] M. Macák, L. Daubner, M. F. Sani, and B. Buhnova, "Process mining usage in cybersecurity and software reliability analysis: A systematic literature review," *Array*, vol. 13, p. 100120, 2022.

[33] M. Müller and P. Ruppel, "Process mining for decentralized applications," in *DAPPCON*. IEEE, 2019, pp. 164–169.

[34] H. Kwon and D. Kim, "A method for churn analysis of new users of mobile games using process mining," *ICIC Express Letters*, vol. v. 7, n. 8, 2016.

[35] S. Kim and D. Kim, "Analyzing mobile application logs using process mining techniques: An application to online bookstores," *ICIC Express Letters*, vol. v. 9, n. 6, 2013.

[36] Y. C. B. Park, I. Cho and W. Lee, "A log analysis of smartphone application usage: Focusing on domestic iphone users," *Journal of the HCI Society of Korea*, vol. v. 2011, n.1, 2011.