Research Proposal

Mining software development projects

Author:
Saimir Bala, M.Sc., h1454574
Darwingasse 9/7, 1020 Vienna, saimir.bala@wu.ac.at

Supervisor:
Prof. Dr. Jan Mendling, jan.mendling@wu.ac.at

Date of Submission:
July 19, 2018

Institute for Information Business, Department of Information Systems and Operations
Vienna University of Economics and Business, Welthandelsplatz 1, 1020 Vienna, Austria

DEPARTMENT FÜR INFORMATIONS-VERARBEITUNG UND PROZESS-MANAGEMENT DEPARTMENT OF INFORMATION SYSTEMS AND OPERATIONS

Abstract

Software development is a highly flexible endeavor. Yet, software development projects have specific requirements about quality, costs and time to deliver. Therefore, there is need for monitoring and controlling the software development process. This problem is tackled separately by three different disciplines, but existing approaches still fail at delivering process knowledge about software development that can be easily understood by managers. This dissertation develops novel methods and algorithms that can be used to extract process knowledge from software development trace data. As a result, it can be positioned as a bridge between the process mining discipline and the area of software engineering. From a research point of view, the outcomes of this dissertation are a building block to further cross-fertilize research streams on process mining and software engineering. From a practical point of view, the outcomes of this dissertation allow the project manager to obtain a more familiar means of analysis on software development such as a process view.

Contents

1 Introduction

Software development is a process that involves creativity. Yet, practical software development projects are executed with specific requirements on time, budget and quality. In order to fulfill these requirements the development process must be monitored. For example, it is important to know what type of work is being done at a certain moment in time and by whom. This is for instance the case for large development projects where coordination mechanisms emerge spontaneously among developers who want to contribute with their code. These type of projects are difficult to control for several reasons. First, there is no clear understanding in how far a certain piece of code advances the current status of the project. Second, a piece of code written by a developer goes through different stages of code-review and it is difficult to predict whether it will ever be merged with the main source. Third, coordination of work may involve a large number of message exchanges among many developers in forums. Hence, it is not feasible to manually oversee what work is going on at a particular point in time.

Literature has tackled this problem from different angles. In the area of process mining approaches exist that exploit event logs for abstracting a process model. In the area of software engineering, software repositories have been explicitly studied. These type of works provide several metrics that help with understanding various aspects of development. However, process mining approaches only work with event logs where activities are explicitly recorded in the log. This is not the case with software development where data is rather unstructured. Likewise, software engineering approaches lack a perspective about work patterns.

This study addresses the discovery of work patterns from software development event data. To this end, it defines concepts to capture work from software repositories. This allows to construct several discovery techniques that provide information about different aspects of the development process. In this sense, this work provides a bridge between process mining and software engineering. The findings of this work enable project managers as well as software developers to raise transparency about the actual development based on facts. As a result, it enables both understanding the current status and monitoring for potentially unwanted patterns of work.

The rest of this proposal is organized as follows. Section 2 describes the problem, the existing literature, and derives the solution requirements. Section 3 provides background knowledge on the related fields of process mining, text mining, and mining software repositories. Section 4 explains the research method, the dataset and outlines the expected output of the thesis. Section 5 shows completed work so far. Section 6 outlines future steps towards the completion of this dissertation. Section 7 draws the implications for research and presents a dissemination plan.

2 Problem Definition

Software development processes are highly complex endeavors that require the coordination of multiple resources. Compared to standard business processes, they present the following characteristics. First, they involve creativity when executing the tasks, i.e. there exists no strict

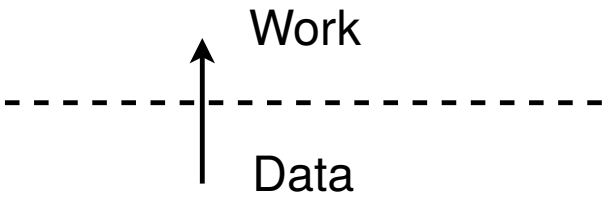

Figure 1: Illustration of the problem: work is reflected by available data

process model that is followed by the developers to produce a new piece of code. Second, they are driven by methodologies, e.g., Rational Unified Process (RUP), Scrum[1], Waterfall, etc. These methodologies constitute guidelines and best practices for project success. Third, they typically make use of software tools such as Issue Tracking System (ITS), Version Control System (VCS) and Integrated Development Environment (IDE). Such tools typically record work activities into log files. Fourth, there is in general no process engine to control the execution. Rather than that, a project manager or a Scrum master has to make sure that tasks are made available to the respective resources. Fifth, it is not trivial to obtain high level information on performance measures such as the burn-down rate, which are resource bottlenecks, what are time requirements for certain tasks, what type of work is actually being done, and how productive are people working in certain tasks, etc. Hence, there is the need for algorithms and tools that help extracting this knowledge.

Software development processes fall into the category of *project-oriented business processes* [BCM+15]. That is, they are rather ad-hoc plans performed with limited resources and time, but with a clear goal: develop a software artifact. Unlike classic business processes which are best captured with notations such as Petri nets and Business Process Model and Notation (BPMN), software development processes are more akin to one-time plans, which are usually captured by models such as Gantt and PERT diagrams. The following example captures the essence of a typical software development process in practice, such as for instance in Google LLC [Hen17].

> A new software version or feature needs to be developed. Unquestionably, Google has know-how on software development. However, before running straight into the development phase, the company first makes sure to gather and properly formulate all the requirements. In a Scrum scenario these requirements would be written down as user stories. At a later stage, the project manager needs to plan the time and resources allocated to the respond to project deadlines. He can go through the list of requirements that need to be implemented and assign to an effort estimation value to each of them. With the plan done, the development phase can commence. During the development phase, resources address the task in a creative way, choosing the order of the tasks according to their own knowledge and expertise, until eventually all the tasks are terminated. Every time a resource wants to save their progress issue a so-called *commit* command which creates a new version of the modified files on the

[1] https://www.scrum.org

VCS. Likewise, every time a certain task (which may be worked carried out through by different commits) is completed, it is marked as done in the ITS. Both the commits and the tasks can be commented by the users, respectively to document the changes and raise a discussion for better understanding the task goals.

Several tools are used by software project participants to support their work. Therefore, traces about the overall process are typically available under different repositories and artifacts, e.g., spreadsheets, word processor documents, programming languages files, emails, etc. This makes it cumbersome to obtain knowledge about the overall business process through manual inspection. Also, it is a challenge to automatically *extract* work information from unstructured data such as user comments when working on tasks. For instance, existing solutions (e.g., process mining algorithms) are not able to provide informative results when the data is not available as a structured single event log where activities are explicitly labeled. Likewise, other software tools (e.g. GitHub) limit themselves at providing only process-unaware statistical information (e.g., number of commits in a file).

While obtaining an overarching view on software development is challenging, there are still repositories that we can be exploited to obtain process knowledge. One tool that provides important traces of the development work is VCS. This tool is used to keep track of the different versions of files created by users and to manage their collaboration. Not only they allow to keep track of the different versions, but also allow users to associate a textual comment that describe the changes made. Therefore, a new version of a particular file is created as the result of an activity done by a user. The evolution of these versions, along with information about the users and their comments can be retrieved from these type of tools in the form of semi-structured event logs. It is then a challenge how to discover the business process from events (e.g., lines of code changed, comments, user information) present in these logs.

Figure 2 illustrates such problem scenario. Software development relies on tools like ITS (e.g., JIRA, GitHub Issues) and VCS (e.g. Subversion, Git). One the one hand, ITS is used for tracking the project management aspect. This aspect offers plan related information, such as issues, features, bugs, text, planned and executed tasks, timestamps, stories, and effort estimation (e.g., story points in JIRA). On the other hand, VCS offers information about work traces, such as versions of the artifacts, number and content of commits done by developers, artifacts evolution, users and their comments, and timestamps of each action.

Although different technologies exist in practice (e.g., Subversion, Mercurial, Git), the information contained in the logs can be roughly summarized by Table 1. The table displays an excerpt of a VCS log, i.e., a set of user commits, after having extracted and structured the data according to five attributes. The semantics of the attributes is a follows: *i) Id* is a unique identifier; *ii) Resource* is the resource that issued the commit; *iii) Date* is a timestamp associated to the time and date the commit was stored in the system; *iv) Comment* is a user comment on the changes made; *v) Diff* is low-level information on the difference between the current and the previous version, for each file. Likewise, information about issues from ITS can be extracted and presented in a tabular way. In this case, other attributes are more important. These attributes can be, for example, the status of issues and the conversations that take place around them. Hence,

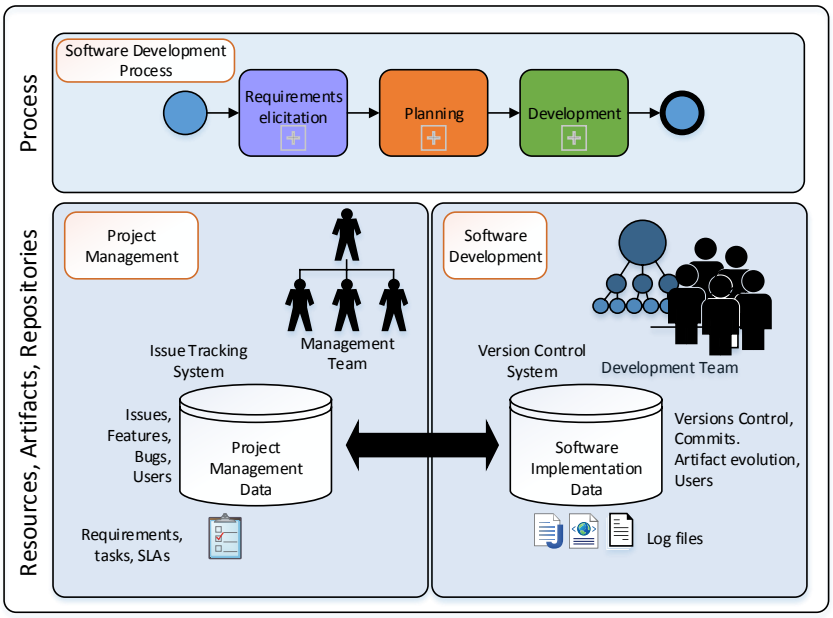

Figure 2: Software development scenario with two repositories used for project management and software development, respectively.

studying the co-evolution of these two repositories can help extracting relevant knowledge about the software development process.

More precisely this work wants to answer to the following research question. **RQ**: *How can we make use of project event data to gather insights about the software development process that are informative to managers?* Because the project managers can benefit from a process view to better analyze hidden aspects of the software development process, this work takes a process mining stance on the problem. Therefore, the main research question is broken down into the following four subquestions. Each of them addresses **RQ** with respect to the four perspectives of a process that are useful to discover the event data [vdA16].

RQ1. Mining the time perspective. *How can we use project event data to extract information about the temporal aspect of activities?* For example, an answer to this question would look like: the development activity has a duration of 2 weeks on average, the average time of task creation is 15 minutes, etc.

RQ2. Mining the case perspective. *How can we use project event data to extract information about the case perspective?* For example, an answer to this question would look like: all the bugs are solved in a 3 steps iteration, or a quality piece of code takes a conversation with 3 people and is successfully merged into the main branch after 1 week, etc.

RQ3. Mining the organizational perspective. *How can we use project event data to extract information about the organizational perspective?* For example, an answer to this question would

Table 1: An excerpt of a VCS log data

Id	Resource	Date	Comment	Diff
1	John	2017-01-31 12:16:30	Create readme file	diff –git a/README.md b/README.md @@ -0,0 +1 @@ +# StoryMiningSoftwareRepositories
2	Mary	2017-02-01 10:13:51	Add a license	diff –git a/README b/README @@ -1,0 +2,3 @@ +The MIT License (MIT) + +Copyright (c) 2015 Mary+
3	Paul	2017-02-02 16:10:22	Updated the requirements.	diff –git a/README.md b/README.md @@ -1,4 +1,5 @@ + # string 1, string 2, string 3 diff –git a/requirements.txt b/requirements.txt @@ -0,0 +1 @@ +The software must solve the problems
4	Paul	2017-02-02 15:00:02	Implement new requirements	diff –git a/model.java b/model.java @@ -1,9 +1,10 @@ +public static methodA(){int newVal=0; @@ -21,10 +23,11 @@ + "1/0",,"0/0", diff –git a/test.java b/test.java @@ -0,0 +1,2 @@ +//test method A +testMethodA()

look like: the software development is carried out by a team of 4 people, the actual user roles in the company are developer and tester, etc.

RQ4. Mining the control-flow perspective. *How can we use project event data to extract information about the control-flow perspective?* For example, an answer to this question would look like: the testing is always done before development, or while new features are worked on, also new requirements are created, etc. Note that, differently from **RQ1** this question focuses on the logical connection and order of activities.

3 State of the Art

The described problem has been partially addressed in three research areas: *i)* Process Mining (PM); *ii)* Mining Software Repositories (MSR); and *iii)* Text Mining (TM). The following sections introduce these fields along with their contribution to the research question and limitations.

3.1 Process Mining

3.1.1 Description of the field

Process mining is a discipline that emerged in the last decade. The goal of this discipline is to provide fact-based insights and support process improvement. 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 [vdA16]. Compared to existing Business Intelligence (BI) technologies, process mining techniques go beyond the calculation of simple Key Performance Indicators (KPIs). Instead, by building on model-driven approaches, they provide means to gain transparency on several aspects of the end-to-end business process. More specifically process mining techniques can infer models from event logs, which inform about the diverse aspects of a business process. As defined in [vdA16], main *perspectives*[2] of a business process are the following.

- The *time perspective* aims at analyzing time and frequency of process events. The focus is to use time information for performance analysis, such as bottlenecks, resource utilization, prediction of workload, remaining running time of process instances, etc. Note that the output of this method is not necessarily a process model, but rather metrics that can be integrated with process models.
- The *case perspective* aims at identifying properties of process cases. A process case is one execution of a process from start to end, e.g. from the moment a customer order a product to its delivery. However, there can be other ways to determine cases, e.g., the evolution of a data artifact, such a for instance the fulfillment of monthly report of the orders.
- The *organizational perspective* aims at analyzing the event log to gain transparency on the resources involved in the process. The focus of this perspective is to help understanding which are the people and systems in the process and how are they related in terms of roles, hierarchy, handover of work, privileges, access-control, social network, etc.
- The *control-flow perspective* aims at analyzing the different variations of the process, i.e., in which order its constituting activities are carried out in real life. The focus of this perspective is on characterizing all possible process paths. Outcome typically results in models expressed as Petri nets, BPMN, event-driven process chain (EPC), Unified Modeling Language (UML), etc.

Process mining is becoming and established field with considerable number of algorithms from academia and a many industry tools such as Celonis[3], Disco[4], minit[5], LANA Process

[2]Note that the different perspectives are partially overlapping. Nevertheless, a good reason to adopt them is because they are widely used in the Business Process Management (BPM) community.

[3]https://www.celonis.com

[4]https://fluxicon.com/disco

[5]https://www.minit.io

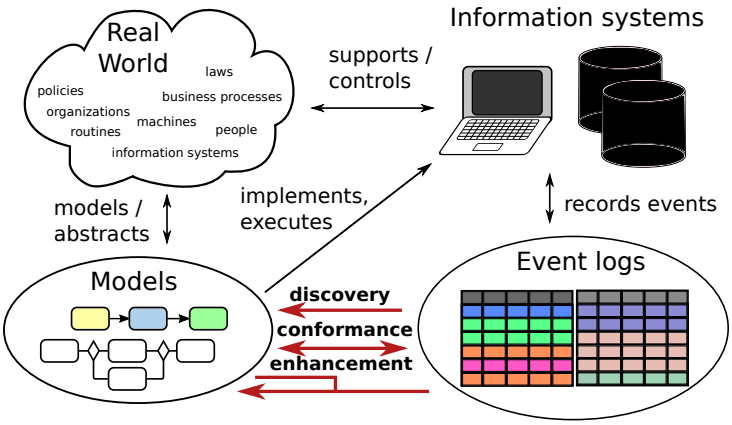

Figure 3: The process mining framework Adapted from [vdA16].

Mining[6] and many more. Mining algorithms are developed every year to deal to mine not only process workflow, but also other perspectives, i.e. organizational, data, etc. Figure 3 illustrates process mining research and how it relates event data from real world and business process models. There are three types of process mining, namely *i)* process discovery; *ii)* conformance checking; and *iii)* enhancement. This proposal focuses more on the discovery part, i.e., take an event log as an input and abstract process patterns from it.

3.1.2 Contribution to the research questions

Several works in the area of PM have tackled the problem by transforming it into a PM problem. Consequently, approaches have been developed to preprocess VCS data such that process mining techniques can be applied, and hence, a business process can be derived from the log data. In this group, Kindler et al. [KRS06a, KRS06b] developed an algorithm for extracting software processes that are mapped to Petri Nets. Activities, which are not explicit in the logs, are discovered from their input and output artifacts. However, strong assumptions are made on the filenames as well as on the software process lifecycle. Rubin et al. in [RGV+07] addressed the problem of engineering processes that are not well documented and are usually unstructured. They provided a bridge from Kindler et al.'s approach to ProM [vDdMV+05] in order to mine different process perspectives, such as performance social network analyses. Rubin et al. [RLvdA14] applied process mining to the touristic industry and obtained user processes from web client logs pursuing the goal of improving the software system by analyzing the underlying process. Poncin et al. [PSvdB11] developed the FRASR framework for preprocessing software repositories to transform the VCS data to logs that conform to the process mining event log meta model [vDdA05] as utilized in ProM [vDdMV+05]. However, these approaches disregard the single-instance nature of project-oriented business processes and treat them as procedures that can be repeated over time.

[6] https://lana-labs.com/en

Process mining techniques are related to all the research questions of this thesis. Especially, they help with addressing **RQ4**, i.e. finding the control flow of software development activities.

3.1.3 Limitations

While providing interesting insights, these contributions leave out many important aspects of software development projects, such as trying to understand whether the process was done according to the organization plan. Especially, they are limited to either simply display one perspective of the process [SvdA07] or transform the events from software repositories into the standard eXtensible Event Stream (XES) format which can be mined by tools like ProM. In this case, existing methods only take into account high level information (such as the type of file) to label events accordingly. Textual information is also taken into account [RGV+07] but often limited to use of a dictionary for identifying keywords. Moreover, fine granular information on the amount of change and the comments of commit messages have not been exploited enough by existing literature.

This works aims to use the strengths of process mining techniques on discovering informative models. At the same time, it aims to go beyond the current limitations of these techniques to handle fine granular (e.g., amount of changes) and unstructured data (e.g. user comments).

3.2 Text Mining

3.2.1 Description of the field

Text mining refers to the process of deriving information from natural language text. It relates to data mining in the aspect that both strive to extract meaningful information from raw data. However, data mining is characterized as the extraction of implicit, previously unknown, and potentially useful information from data, whereas with text mining the information to be extracted is clearly and explicitly stated in the text [Wit04]. Text mining can be regarded as going beyond information access to further help users analyze and digest information and facilitate decision making. There are also many applications of text mining where the primary goal is to analyze and discover any interesting patterns, including trends and outliers in text data. Although text mining is mostly about Natural Language Processing (NLP) [JM14], it embraces also applications that go beyond. For example, it analyzes linkage structures such as the citations in the academic literature and hyperlinks in the Web literature, both useful sources of information that lie outside the traditional domain of NLP.

3.2.2 Contribution to the research questions

NLP based approaches have been used to identify process cases and activities. Contributions in this group the have focused on process discovery. In [dARGSB09] a model is discovered from group stories. Work from [FMP11] reaches 77% of accuracy in reconstructing process models from text. In [dNIT+12] legacy systems code is analyzed to infer business process rules

and activities. The work of [DM13a] uses NLP to aid the extraction of artful processes from knowledge workers emails.

There are also works that help with unstructuredness, although they do not take a process perspective. Maalej and Happel [MH10] use NLP for automating descriptions of work sessions by analyzing developers' informal text notes about their tasks. Developers are then classified into two classes based on their behavior: developers who use problem information to refer to their current activity and developers who refer to task and requirements. Kouters et al. [KVSvdB12] developed an identity merging algorithm based on Latent Semantic Analysis (LSA) to disambiguate user emails. Licorish and MacDonell [LM14] mined developer comments to understand their attitudes.

Role discovery has also seen contributions. A number of algorithms have been developed to mine roles from role-based access-control (RBAC) systems alone ([LHY+15, FBB13]) or combining their data with process history logs, as in Baumgraß et al. [BSWS12]. A survey of existing techniques and algorithms can be found in [MSVA16].

This body of works indeed suggests that valuable process insights can be obtained from unstructured data. Therefore, text mining research serves and a preprocessing technique to facilitate process extraction. Especially, it helps as follows *i)* **RQ2** – extracting case characteristics from conversations in user forums; *ii)* **RQ3** – extracting resource roles from user comments in commit messages; *iii)* **RQ4** – extracting activities out of coordination message exchanges.

3.2.3 Limitations

Text mining techniques do not natively support extraction of processes, but they can be used to help addressing information extraction from unstructured data. TM approaches focus on obtaining structured information from unstructured textual data, and mainly uses NLP [WBMT99]. Works that use NLP can be found in both PM [vdA16] and MSR [CTH16]. In the BPM [DRMR18] area, NLP techniques have been used to understand process activities [Leo13, MLP14] and analyze software processes under a knowledge-intensive perspective [dSB11, RdBS17]. Likewise, in the MSR area, NLP has been used as an information extraction tool to obtain informative metrics from a software engineering perspective [THB14, CTH16].

3.3 Mining Software Repositories

3.3.1 Description of the field

Mining software repositories (MSR) research has emerged in the last decade from the software engineering field. It focuses on analyzing patterns and trends in software. Main topics in MSR are about software quality and metrics, visual software analytics, bug analysis, communication among project members, defect tracking, predictions about software development, and software evolution. Works in MSR often use techniques from data mining, text mining and statistics. Typically, these methods focus on analyzing dependencies in the structure of the software artifacts. There are some existing works that can be taken into account as a starting point for

understanding the development process. These works focus principally on the users and the artifacts, mining co-evolution or co-change of project parts [ZRDvD08, DLL09] and network analysis of file dependency graph based on commit distance [ZN08, ABDZ09, YSX13]. Hidden work dependencies are mentioned as *logical dependencies* [OSGdS11]. Also techniques for trend analysis [RHL15] and inter-dependencies between developers [LBGL16] are proposed. All these techniques allows to obtain several metrics and models from software repository data.

3.3.2 Contribution to the research questions

MSR focuses on software engineering aspects, like code quality metrics, modularization, code complexity, user networks, and other important metric. In general, methods from MSR provide useful dependency analyses of the repository structure. Contributions in this area focus on the users, the artifacts and the repository evolution [ZRDvD08, DLL09], and network analysis of file dependency graph based on commit distance [ABDZ09, YSX13]. Liguo and Ramaswamy [YR07] use a hierarchical clustering based on user interactions to identify two categories of users: *core member* and *associate member*. Core members are those users whose interaction frequency is higher than a given threshold. Associate members are instead users whose interaction frequency is below the threshold. Alonso et al. [ADG08] use a rule-based classifier that maps file types onto categories and hence each author who modified a file is linked to the files' category. Gousios et al. [GKS08] classify developers contribution based on lines of code (LOC) changes and infer activities from them. Begel et al. [BKZ10] developed the Codebook software tool a utility for finding experts. They use a social network approach that combines sources from people, artifacts, and textual references to other people. Ying and Robillard [YR14] study developer profiles in terms of their interaction with the software artifacts to understand how they modify files and to further recommend changes based on history from VCS logs. Füller et al. [FHHM14] investigate user roles in innovation-contest communities. They use quantitative methods to analyze user activity logs and interpretative to categorize qualitative comments into classes. Also techniques for trend analysis [RHL15] and inter-dependencies between developers [LBGL16] are proposed.

The above-mentioned techniques inform this research on relevant metrics about software development. They are relevant to the following research questions. *i)* **RQ1** – through gathering times and durations of development activities; *ii)* **RQ2** – through helping the extraction of artifact evolution; *iii)* **RQ3** – though helping extraction of social networks of users.

3.3.3 Limitations

Limitations of MSR works derive from its software engineering orientation. In fact, none of these works aims at extracting knowledge about the actual process. Extraction is typically limited to engineering metrics and rarely goes beyond mining the bug lifecycle [PSvdB11] when it comes to process discovery. This works assumes that both the communities of MSR and PM could benefit from research on mining process models out of software development activities. The former would gain better perspectives and easier understanding of the engineering activities.

The latter would extend its range towards a new domain. In this sense, this thesis serves as an attempt to bridge the above-mentioned fields.

3.4 Summary

This research combines ideas from the above mentioned areas to devise algorithms for mining the software development process. This section defines the contribution of the related fields to the four research question. Table 2 lists summarizes the most relevant works in the literature and classifies them in relation the addressed research questions.

Table 2: Classification of existing literature addressing the various aspects of mining the development process

	R1	R2	R3	R4
PM	Dotted Chart [SvdA07]	Decision mining [RvdA06]	Visualization techniques [BS13] Organizational mining [SvdA08] [SCJM15]	Bug fixing [PSvdB11], Workflow fragments [KRS06a, KRS06b]
TM	Information extraction [CL96]	Topic models [CTH16] Theory-generating case studies [LBGL16]	Social network [BGD+06] Survey [BKZ10] [dASB10]	Speech acts [DM13b] [CRBS18] Natural language processing [FMP11]
MSR	Time series [RHL15] [HMCX14], Statistical analyses [OSGdS11]	Network analysis [DLL09], [ZN08], Language Models [AS13]	Email analysis [BGD+06], Role identification [YR07]	Exploratory studies [GPvD14]

4 Research Design and Methodology

This section presents the research method. It also describes the dataset and outlines the expected research outcome.

4.1 Research Method

Information systems research is an interdisciplinary field of study that uses theories from social sciences, economics, and computer science. The field can be divided into two complementary

paradigms: *behavioral science* and *design science*. Behavioral science aims at developing and justifying theories in order to explain or predict information systems phenomena [Gre06]. Design science focuses on the creation and evaluation of innovative design *artifacts* [HMPR04]. Figure 4 illustrates the design science approach as a process [PTRC08].

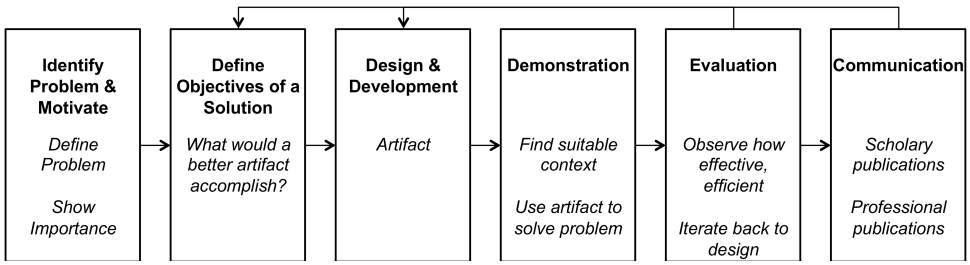

Figure 4: The Design Science Research process, adapted from [PTRC08]

This proposal foresees the development of four different artifacts. These artifacts are interconnected to one another in that each of them tackles a different perspective of the overarching software development process. More specifically, each artifact is devised following the Design Science Research (DSR) approach. These artifacts are useful useful to inform the mentioned approach to theory building [BSS18].

Research rigor and validity are ensured by evaluating the artifacts with the Framework for Evaluation in Design Science Research (FEDS) [VPHB16]. FEDS provides strategies for evaluating DSR artifacts. More specifically, it takes into account two dimensions *i)* the functional purpose of the evaluation (formative or summative); and *ii)* the paradigm of the evaluation (artificial or naturalistic). Moreover, it provides four steps for choosing an appropriate evaluation strategy *i)* explicate the goals of the evaluation; *ii)* choose the evaluation strategy or strategies *iii)* determine the properties to evaluate; and *iv)* design the individual evaluation episode(s).

Next, the evaluation strategies for each designed artifact are provided.

Artifact A1.

Design objectives of **A1** are *(i)* has to be applicable, *(ii)* usable, *(iii)* simple and, *(iv)* provide overview as well as detailed view. The following features are included: *(i)* time information, *(ii)* duration of activities, *(iii)* structure of the project, and *(iv)* roll-up and drill-down on granularity of events. Evaluation goals are *(i)* rigor, *(ii)* uncertainty and risk reduction, *(iii)* ethics, and *(iv)* efficiency. They are respectively addressed as follows. Rigor is assured by the design science approach. Uncertainty involves technical infeasibility and unavailability of the data. Infeasibility risk is reduced by starting from the design of small artifacts with basic features and incrementally improve. Unavailability risk is reduced by replication of the dataset locally. An ethics problem may regard the "big-brother is watching you" effect on people. This will be solved considering only data from partners and public repositories. Efficiency is evaluated by

measuring whether all time patterns are found in a reasonably short time.

This artifact is evaluated through a Quick & Simple strategy (see [VPHB16]), i.e., the technique is directly evaluated with real data. This will include three iteration episodes *(i)* initial design of Gantt chart (formative) *(ii)* evaluation of successful identification events, their duration and project structure; and *(iii)* run of the tool on real projects and provide a visualization (summative).

Artifact A2.

Design objectives of **A2** include applicability, simplicity and providing comparison of different cases. The following features are designed: *(i)* difference between work instances; *(ii)* measure of work: amount of changes; *(iii)* handle unstructured data from comments; *(iv)* evolutionary analysis of files; and *(v)* uncover work dependencies. Evaluation goals are as follows. Rigor is assured by the iterations. Uncertainty and risk reduction consist in: technical infeasibility – addressed through development by smaller iterations, data unavailability – addressed by local replication of data, and simplicity of results – addressed by exploiting user comments and derive informative process labels. Ethics problems do not arise as the data for this artifact is already public on GitHub. Efficiency is evaluated by finding all work dependencies in a reasonable time.

This artifact is evaluated through a Technical risk & Efficacy strategy (see [VPHB16]), i.e., the technique is firstly evaluated with a toy example, then revised and applied to a large number of software development projects. Five iteration episodes are included: *(i)* initial design of artifact and results visualizations(formative); *(ii)* revision and choice of final visualization; *(iii)* evaluation of efficacy on retrieving features; *(iv)* run the technique on toy example (artificial evaluation), validate efficacy and usefulness, revise artifact; and *(v)* run the technique on real data sets from GitHub projects (summative and naturalistic evaluation).

Artifact A3.

Design objectives of **A3** include applicability and correct classification of roles. The following features are designed *(i)* determine roles of users; *(ii)* handle user comments; and *(iii)* automatically classify resources. Evaluation goals are as follows. Rigor is assured by the iterations. Uncertainty and risk reduction regard classification. Incorrect resource classification risk is reduced by using different training set data from industry partners and obtaining feedback. An ethics risk is about obtaining information about people's work. This is mitigated by NDAs signed by the parties involved. Efficiency is evaluated in terms of precision and recall.

This artifact is evaluated through a Quick & Simple strategy (see [VPHB16]), i.e., the technique is evaluated with real data from industry partners. Three iteration episodes are included: *(i)* initial design of the artifact voted to a simple and structured representation of the data (formative) *(ii)* exploration and selection of the best features and classifiers. *(iii)* evaluation of results with real data from partners and feedback (summative + naturalistic).

Artifact A4.

Design objectives of **A4** are its applicability to pull requests and its functionality to provide informative and reliable process models about specific work patterns. The following features are designed. *(i)* handle comments from forum conversations, and *(ii)* discover a process model. Evaluation goals are considered as follows. Rigor is assured by the iterations. Uncertainty and risk reduction the following *(i)* activities of business processes are not mapped correctly – mitigated by manual annotation; and *(ii)* the resulting model is not informative – mitigated by statistical techniques voted to cluster traces into significant groups. Ethics problems do not arise because only data from open source repositories will be used. Efficiency is measured by the extent to which the artifact can deal with complex projects quickly.

This artifact is evaluated through a Technical risk & Efficacy strategy (see [VPHB16]), i.e., the technique is firstly evaluated with an initial well-known dataset, then revised and evaluated with other projects. Three iteration episodes are included: *(i)* initial exploratory analysis on the applicability of process mining techniques on pull requests (formative); *(ii)* evaluation of process mining methods and assessment of statistical significance of results; and *(iii)* mine process models that are statistically significant and analyze the idea generation patterns (summative + naturalistic).

4.2 Data Collection

This thesis includes the collection of and generation of datasets from real life software projects. These project data were collected both from industrial partners and from Open-source Software (OSS). Table 3 summarizes the available data. Logs from industrial partner are concerned with specific development activities (e.g., building railway interlocking-system software) and include a sufficient number of events that cover one software release. Logs from OSS were manually extracted by GitHub repositories. A tool has then been devised to parse these data and use them to populate a database. An original dataset is represented by Github pull requests. This dataset contains a list of manually annotated pull requests using the coding scheme by [MM16]. An additional output of this thesis will be the creation of a larger dataset using a machine learning approach that is able to automatically categorize pull requests in the classes given by [MM16].

4.3 Expected Outcome

In addition to an original dataset, this doctoral thesis is expected to produce outcomes that help the project manager analyze his software development processes. To this end new approaches in the form of artifacts [PTRC08] will be devised. These artifacts will serve as proofs-of-concept for the applicability of the research. In particular, the artifacts must address each of the four process aspects. Therefore, the outcome is categorized according to the research questions derived in Section 2 as follows.

- **Artifact A1: Mining the Time Perspective (RQ1)**. Novel technique that allows for gaining

Table 3: Available dataset

Log	Description
SHAPE	Industry VCS from software development in the railway domain
Github	Logs from real world OSS development
Github pull requests	Manually annotated pull requests from one real life open source project
Jira	Log data from ITS from industry partner
Asana	Generated dataset from project management tool of industrial partner
Kibana	Event traces from distributed process-agnostic enviroment used to handle communication among several processes

transparency on the time perspective of software development work. For instance, and artifact that explicits what are the durations of tasks and when are actual tasks executed.

- **Artifact A2: Mining the Case Perspective (RQ2)**. Novel technique that allows for gaining transparency on the case perspective of software development work. For instance, an artifact that explicits how the different cases are handled and whether there is any dependency of work that might influence the case execution.
- **Artifact A3: Mining the Organizational Perspective (RQ3)**. Novel technique that allows for gaining transparency on the organizations aspect of software development work. For instance, an artifact that explicits which are the actual roles software developers are covering and which is the organizational network.
- **Artifact A4: Mining the Control-Flow Perspective (RQ4)**. Novel technique that allows for gaining transparency on the control-flow perspective of software development work. For instance, an artifact that allows for abstracting from data which is order of activities that are executed to accomplish a certain task or goal in software development.

Figure 5 shows the overall information coming from combining the four perspectives into a Gantt chart that is more informative to managers.

5 Preliminary Results

Next, I present efforts done towards addressing the requirements presented in Section 2.

5.1 A1. Mining the Time Perspective (RQ1)

Mining the time perspective from a software repository is defined as extracting temporal process knowledge. This problem relates to monitoring whether the process was executed respecting a

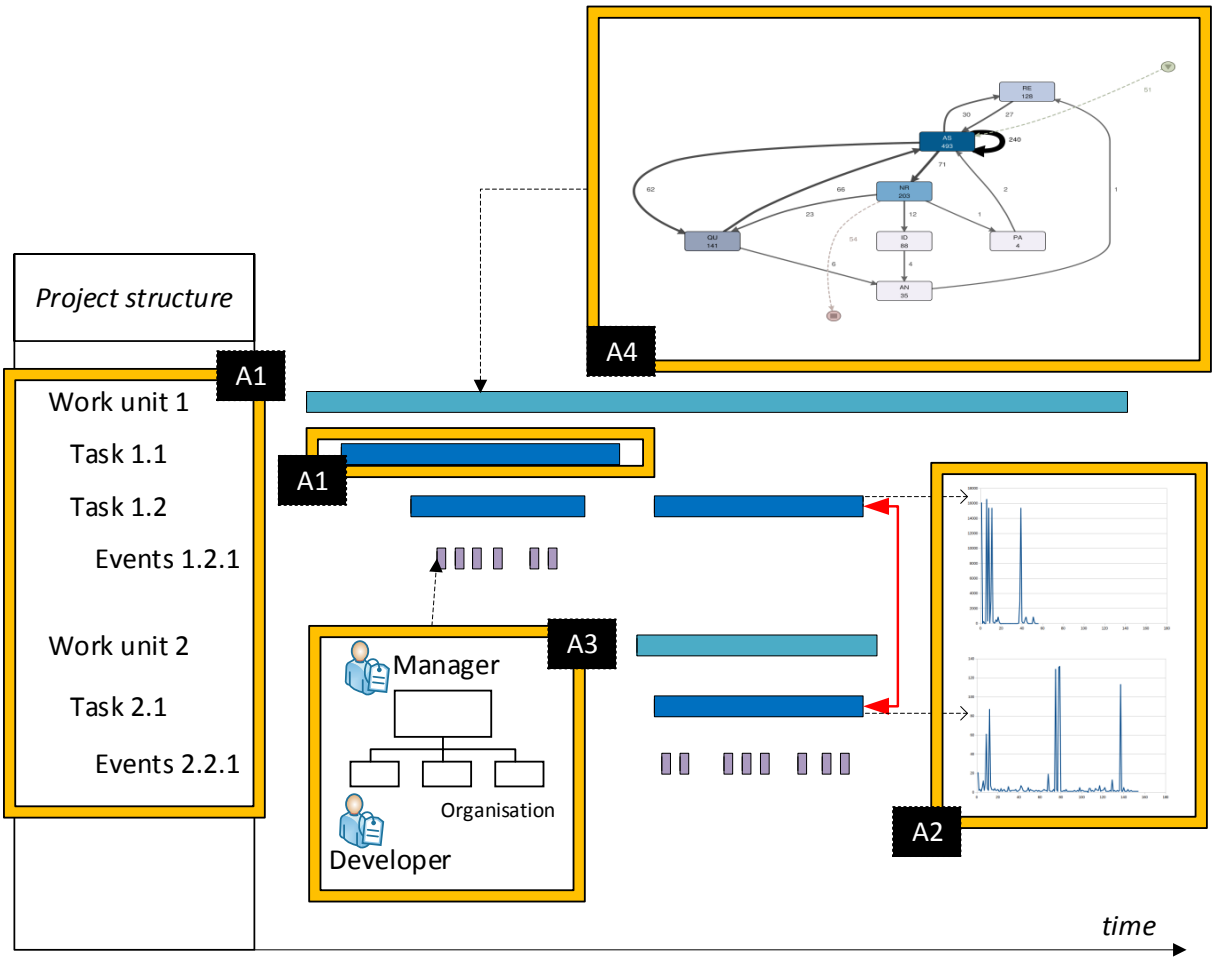

Figure 5: The empirical Gantt chart of software development. The four process perspectives combined.

predefined plan. Typically the process involves various actors which track their work through the use of VCS. Often, the plans are not represented in standard process notation. Rather, Gantt or PERT charts are used.

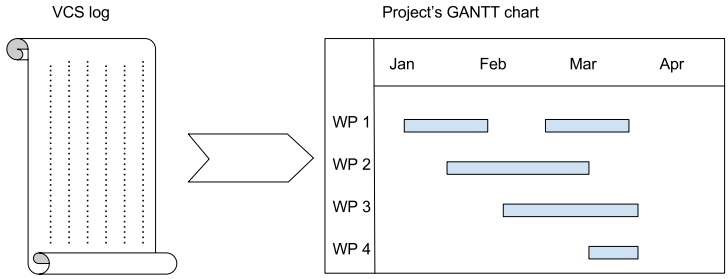

Figure 6: Mining the time perspective of software development into a Gantt chart

Challenges are *i) time approximation* (i.e., when the activity really started, compared to when it was registered in the log); *ii) granularity* (i.e., be able to switch from a detailed view of the single events and a coarse-grained view of the overall project); and *iii) coverage* (i.e., how much work effort was put within the duration of the activity). Figure 6 depicts the idea of mining a Gantt chart from VCS logs. The output is presented in a way that is informative to project managers. Details about the technique can be found in [BCM+15].

5.2 A2. Mining the Case Perspective (RQ2)

The identification of process cases is not an easy task when it comes to software development data. However, a highly informative case candidate can be considered the data artifact itself. According to [vdA16], cases can also be characterized by the values of data elements. In line with this, it is possible to devise techniques that study the *artifact evolution*. Although it can be input to process mining techniques, this evolution can also be analyzed as a time series. An example of such evolution is given in Figure 7, which shows the LOC changes over time.

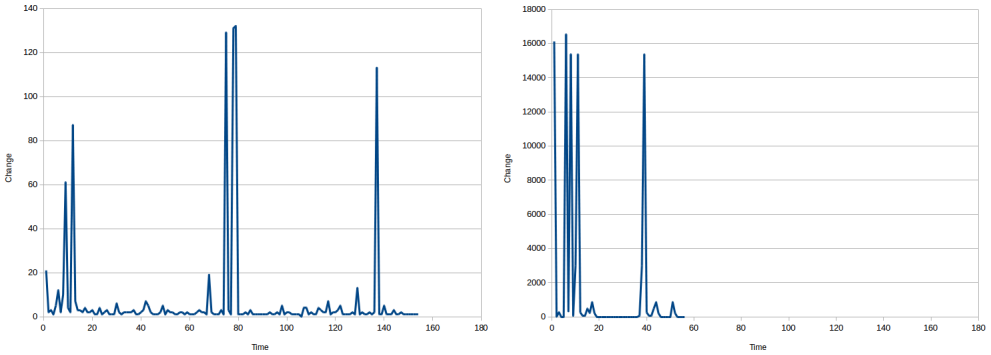

Figure 7: Evolution of files in a version control systems: LOC changes over time

Challenges related to the artifact evolution pertain *prediction* of plateaus and *pattern recognition*. The results reflect work patterns over the artifact. For example, when a data file is not being modified anymore, its time series has a plateau and it can be interpreted that the document is now ready for release. A relevant problem in software development is bad modularization. Especially, the dependency of two artifacts on one another is considered a bad practice. Two time series can be compared together and file dependencies can be found that reflect work coupling. This is equivalent to finding dependence between two processes which might have been designed for different purposes.

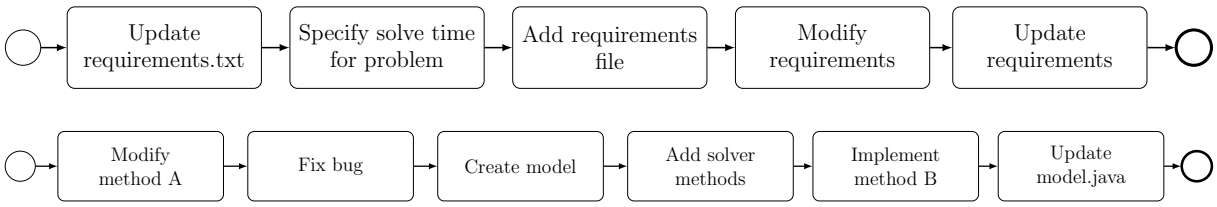

Figure 8: Processes of two work-dependent files. They may seem different but they co-evolve in time and space, i.e., they describe the same *case*.

We have devised a proof-of-concept in [BRd+17]. Figure 8 shows a possible outcome of the technique. The result is obtained by identifying couples of files with similar evolution and mining a business process from the comments. As the result is aggregated, several comments have been combined together and mined as a story. Among other things, the technique allows to profile existing software development projects.

5.3 A3. Mining the Organizational Perspective (RQ3)

Software development projects are knowledge intensive, thus involve creativity and flexibility. Project participants are free to tackle development tasks according to their expertise and ideas to solve ad-hoc problems. De facto, members may behave according several roles in the organization. Therefore, the discovery of *de facto* roles within a software project may give important insights on the actual project. Role discovery can help with monitoring existing projects in two ways. First, emerging roles can be analyzed in order to have a better skills profiling of the resources. Second, emerging roles can be used to check whether resources are breaking contractual agreements or norms.

We have developed an approach for tackling the problem of mining the organizational perspective in software development projects. The approach provides information about the actors involved in the business process and their relations. In substance, it provides automatic classification of resources into company roles (e.g., developer, tester, etc) based on the comments of project participants. It works on VCS logs and provides information about the people, their roles, other systems involved, the organization hierarchy, the social network, and resource profiling. Figure 9 shows an example of resource profiles automatically inferred from VCS

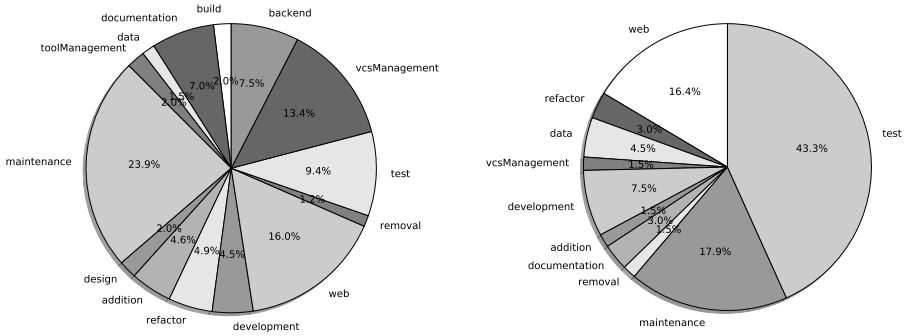

Figure 9: Profiles of a developer and a tester, from left to right.

comments.

Challenges of using NLP concern the *jargon* used by software developers. They often use technical terms, very short phrases, or a number of codes and hyperlinks. These makes NLP techniques score low results even on simple tasks like sentence parsing, or name-entity recognition. Therefore, approaches should take into account several factors when mining for the organizational perspective. The result allows the manager to have measurable information about the resources. For example, do two people work better together and how can we build the best team? Details of the approach to extract resource profiles can be found in [AAT+16].

6 Next Steps and Finalization of the Thesis

6.1 A4. Mining the Control-Flow Perspective (RQ4)

Knowledge of the control flow of a software development process can be extracted from VCS and ITS. One important mechanism widely used in software development for collaboration are *pull requests*. A pull request is issued every time a developer wants to contribute to the main source with a new piece of code. Pull requests trigger communication among different people and may help shedding light into problems, learn new things and obtain new ideas on the existing software. It is interesting for project managers and developers to better understand factors and behavior that makes a pull request successful.

In ongoing work, we are investigating the relation of the pull request process and surrounding factors such as the generation of new ideas or change of behavior. We have obtained a data set of pull requests spanning over two years from an real world repository. User conversations have been manually annotated with codes that classify them into categories of comments. We are current able to discriminate whether a comment is a new idea, an assumption, a merge that closes a pull request, etc. Considering the pull request identifiers as cases and the codes as activities, we are able to extract the process of the pull requests. It is interesting to compare how the conversation (i.e., behavioral perspective) unfolds before and after a pull request is merged. We plan to explain these differences by also taking into account further information from text

such as emotions.

The challenge of mining the control-flow usually pertains the completeness of data and the case and activity identification. While PM contribution have been focusing mostly on the control-flow perspective, techniques from MSR hardly go beyond mining the bug lifecycle [Akb18]. The nature of software development process makes it difficult to recognize recurrent activities when only VCS are analyzed.

6.2 Finalization of the Thesis

The expected result for the above-mentioned work is an artifact that is able to explicitly show what are process patterns that lead to the generation of an idea. This will serve as a proof-of-concept that event data from projects can be used to increase understanding of process flow perspective. The overall thesis will be finalized by evaluating the results both against the real world dataset and project managers. There is ongoing work in which a questionnaire has been answered by industry partners. Questions pertains activities, tools and habits within a software development company. These information can be used to evaluate the artifacts developed by this thesis.

7 Dissertation Relevance and Publication Plan

7.1 Implications for Research

This proposal presents research on mining the software development process. Approaches towards discovering the perspectives of time, resources and cases have been developed and the results suggest that obtaining process knowledge from software repositories brings new useful insights for the project managers. Ongoing work aims to complete the research by tackling the problem of discovering the flow-perspective of the software development process. Usefulness will be evaluated through user studies. This research extends the process mining field towards its adoption on software repositories. Project managers would benefit from this research by having a process perspective on their ongoing projects through visual models and diagrams, thus overcoming existing flat approaches based on simple indicators such as burndown charts or change plots offered by existing tools.

7.2 Work Plan and Dissemination

Figure 10 summarizes the work plan based on the artifacts to be build for addressing the four research questions. In other words, the thesis work is currently in its final stage, which involves the creation of artifact **A4** to address the corresponding research question **RQ4**.

Target venues for communicating the results are BPM and software engineering journals and conferences. So far, the following publications have been achieved that directly address the research questions concerning time, organization, case and control-flow, as defined in Section 2.

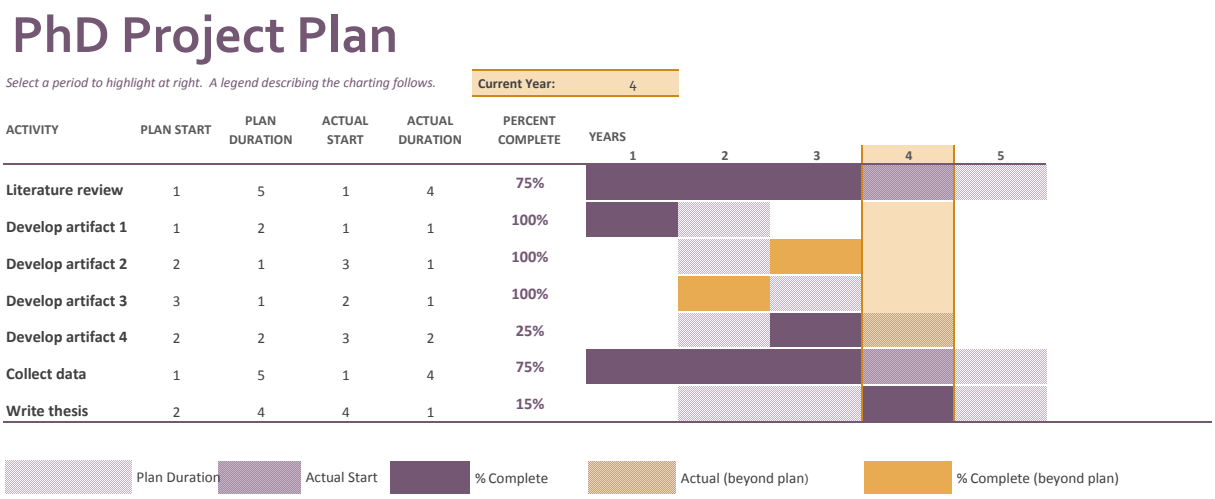

Figure 10: Gantt chart of PhD Plan

Mining the Time Perspective (**RQ1**):

- **Bala, S.**, Cabanillas, C., Mendling, J., Rogge-Solti, A., and Polleres, A.: *Mining Project-Oriented Business Processes*. In Hamid Reza Motahari-Nezhad, Jan Recker, and Matthias Weidlich, editors, BPM 2015, Innsbruck, Austria, volume 9253 of Lecture Notes in Computer Science, pages 425–440. Springer, 2015. [BCM+15]

Mining the Case Perspective (**RQ2**):

- **Bala, S.**, Revoredo, K., de A.R. Gonçalves, J.C., Baião, F., Mendling, J., Santoro, F.: *Uncovering the Hidden Co-evolution in the Work History of Software Projects*. In: Carmona, J., Engels, G., and Kumar, A. (eds.) Business Process Management - 15th International Conference, BPM 2017, Barcelona, Spain, September 10-15, 2017, Proceedings. pp. 164–180. Springer (2017). [BRd+17]

Mining the Organizational Perspective (**RQ3**):

- Agrawal, K., Aschauer, M., Thonhofer, T., **Bala, S.**, Rogge-Solti, A., Tomsich, N.: *Resource Classification from Version Control System Logs*. In: Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOC Workshop. pp. 249–258 (2016). [AAT+16]

Publications that address the requirements in Section 2 indirectly.

- **Bala, S.**, Cabanillas, C., Haselböck, A., Havur, G., Mendling, J., Polleres, A., Sperl, S., Steyskal, S.: *A Framework for Safety-Critical Process Management in Engineering Projects*. In: Ceravolo, P. and Rinderle-Ma, S. (eds.) Data-Driven Process Discovery and Analysis-SIMPDA. pp. 1–27. Springer (2015). [BCH+17] – Related to **RQ1** & **RQ3**.

- **Bala, S.**, Havur, G., Sperl, S., Steyskal, S., Haselböck, A., Mendling, J., Polleres, A.: SHAPEworks: A BPMS extension for complex process management. In: CEUR Workshop Proceedings. pp. 50–55 (2016). [BHS+16] – Related to **RQ1** & **RQ3**.

Doctoral consortium and vision paper presenting the concepts of this research proposal.

- **Bala, S.**: *Mining projects from structured and unstructured data*. In: CEUR Workshop Proceedings (2017). [Bal17]
- **Bala, S.**, Mendling, J.: *Monitoring the Software Development Process with Process Mining*. In Boris Shishkov, editor, Business Modeling and Software Design, volume 319 of Lecture Notes in Business Information Processing, 2018 [BM18]

From ongoing work the following publications are expected.

- Case study on tool productivity. Collaboration with researchers from the University of Ljubljana to be submitted to a class A journal on information systems. – Related to **RQ1**, **RQ3** and **RQ4**.
- Process mining pull requests for identifying idea-creation patterns. Collaboration with researchers from Stevens Institute of Technology to be submitted to a class A journal on information systems. – Related to **RQ4**.
- Mining knowledge intensive processes from pull requests. Collaboration with researchers from Federal University of the State of Rio de Janeiro (UNIRIO) to be submitted to a class A conference. – Related to **RQ2** & **RQ4**.

Other work in the BPM area.

- Woliński, B., **Bala, S.**: *Comprehensive Business Process Management at Siemens: Implementing Business Process Excellence*. In: vom Brocke, J. and Mendling, J. (eds.) Business Process Management Cases: Digital Innovation and Business Transformation in Practice. pp. 111–124. Springer International Publishing, Cham (2018). [WB18]

References

[AAT+16] Kushal Agrawal, Michael Aschauer, Thomas Thonhofer, Saimir Bala, Andreas Rogge-Solti, and Nico Tomsich. Resource Classification from Version Control System Logs. In *Proc. - IEEE Int. Enterp. Distrib. Object Comput. Work. EDOCW*, volume 2016-September, pages 249–258, 2016.

[ABDZ09] Pietro Abate, Jaap Boender, Roberto Di Cosmo, and Stefano Zacchiroli. Strong dependencies between software components. *2009 3rd Int. Symp. Empir. Softw. Eng. Meas. ESEM 2009*, pages 89–99, 2009.

[ADG08] Omar Alonso, Premkumar T. Devanbu, and Michael Gertz. Expertise identification and visualization from CVS. *Proc. 2008 Int. Work. Min. Softw. Repos. - MSR '08*, page 125, 2008.

[Akb18] Shirin Akbarinasaji. Prioritizing lingering bugs. *ACM SIGSOFT Softw. Eng. Notes*, 43(1):1–6, 2018.

[AS13] Miltiadis Allamanis and Charles Sutton. Mining source code repositories at massive scale using language modeling. *IEEE Int. Work. Conf. Min. Softw. Repos.*, pages 207–216, 2013.

[Bal17] S. Bala. Mining projects from structured and unstructured data. In *CEUR Workshop Proc.*, volume 1859, 2017.

[BCH+17] Saimir Bala, Cristina Cabanillas, Alois Haselböck, Giray Havur, Jan Mendling, Axel Polleres, Simon Sperl, and Simon Steyskal. A Framework for Safety-Critical Process Management in Engineering Projects. In Paolo Ceravolo and Stefanie Rinderle-Ma, editors, *Data-Driven Process Discovery and Analysis*, pages 1–27, Cham, 2017. Springer International Publishing.

[BCM+15] Saimir Bala, Cristina Cabanillas, Jan Mendling, Andreas Rogge-Solti, and Axel Polleres. Mining Project-Oriented Business Processes. In Hamid Reza Motahari-Nezhad, Jan Recker, and Matthias Weidlich, editors, *BPM 2015, Innsbruck, Austria*, volume 9253 of *Lecture Notes in Computer Science*, pages 425–440. Springer, 2015.

[BGD+06] Christian Bird, Alex Gourley, Prem Devanbu, Michael Gertz, and Anand Swaminathan. Mining email social networks. In *Proc. 2006 Int. Work. Min. Softw. Repos.*, pages 137–143. ACM, 2006.

[BHS+16] Saimir Bala, Giray Havur, Simon Sperl, Simon Steyskal, Alois Haselböck, Jan Mendling, and Axel Polleres. SHAPEworks: A BPMS extension for complex process management. In *CEUR Workshop Proc.*, volume 1789, pages 50–55, 2016.

[BKZ10] Andrew Begel, Yit Phang Khoo, and Thomas Zimmermann. Codebook: discovering and exploiting relationships in software repositories. In *2010 ACM/IEEE 32nd Int. Conf. Softw. Eng.*, volume 1, pages 125–134, 2010.

[BM18] Saimir Bala and Jan Mendling. Monitoring the software development process with process mining. In Boris Shishkov, editor, *Business Modeling and Software Design*, volume 319 of *Lecture Notes in Business Information Processing*, 2018.

[BRd+17] Saimir Bala, Kate Revoredo, João Carlos de A.R. Gonçalves, Fernanda Baião, Jan Mendling, and Flavia Santoro. Uncovering the Hidden Co-evolution in the Work History of Software Projects. In Josep Carmona, Gregor Engels, and Akhil Kumar, editors, *Bus. Process Manag. - 15th Int. Conf. BPM 2017, Barcelona, Spain, Sept. 10-15,*

2017, Proc., volume 10445 of *Lecture Notes in Computer Science*, pages 164–180. Springer, 2017.

[BS13] Anne Baumgrass and Mark Strembeck. Bridging the gap between role mining and role engineering via migration guides. *Inf. Sec. Techn. Rep.*, 17(4):148–172, 2013.

[BSS18] Nicholas Berente, Stefan Seidel, and Hani Safadi. Data-Driven Computationally-Intensive Theory Development. *Inf. Syst. Res.*, 2018.

[BSWS12] Anne Baumgrass, Sigrid Schefer-Wenzl, and Mark Strembeck. Deriving process-related rbac models from process execution histories. In *Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual*, pages 421–426. IEEE, 2012.

[CL96] Jim Cowie and Wendy Lehnert. Information extraction. *Commun ACM*, 39(1):80–91, 1996.

[CRBS18] Júlio Campos, Pedro Richetti, Fernanda Araújo Baião, and Flávia Maria Santoro. Discovering Business Rules in Knowledge-Intensive Processes Through Decision Mining: An Experimental Study BT. In Ernest Teniente and Matthias Weidlich, editors, *Business Process Management Workshops*, pages 556–567, Cham, 2018. Springer International Publishing.

[CTH16] Tse-Hsun Chen, Stephen W Thomas, and Ahmed E Hassan. A survey on the use of topic models when mining software repositories. *Empir. Softw. Eng.*, 21(5):1843–1919, 2016.

[dARGSB09] João Carlos de A. R. Gonçalves, Flávia Maria Santoro, and Fernanda Araujo Baião. Business process mining from group stories. In *CSCWD*, pages 161–166. IEEE, 2009.

[dASB10] Joao Carlos de A.R.Goncalves, F M Santoro, and F A Baiao. A case study on designing business processes based on collaborative and mining approaches. In *Comput. Support. Coop. Work Des. (CSCWD), 2010 14th Int. Conf.*, pages 611–616, apr 2010.

[DLL09] Marco D'Ambros, Michele Lanza, and Mircea Lungu. Visualizing co-change information with the evolution radar. *IEEE Trans Softw Eng*, 35(5):720–735, 2009.

[DM13a] Claudio Di Ciccio and Massimo Mecella. Mining Artful Processes from Knowledge Workers' Emails. *IEEE Internet Comput*, 17(5):10–20, 2013.

[DM13b] Claudio Di Ciccio and Massimo Mecella. Mining Artful Processes from Knowledge Workers' Emails. *Internet Comput. IEEE*, 17(5):10–20, 2013.

[dNIT+12] Gleison S do Nascimento, Cirano Iochpe, Lucinéia Thom, André C Kalsing, and Álvaro Moreira. Identifying Business Rules to Legacy Systems Reengineering Based on BPM and SOA. In Beniamino Murgante, Osvaldo Gervasi, Sanjay Misra, Nadia Nedjah, Ana Maria A C Rocha, David Taniar, and Bernady O Apduhan, editors, *Computational Science and Its Applications*, pages 67–82, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[DRMR18] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. *Fundamentals of Business Process Management, Second Edition*. Springer, 2018.

[dSB11] João Carlos de A. R. Gonçalves, Flávia Maria Santoro, and Fernanda Araujo Baião. Let Me Tell You a Story - On How to Build Process Models. *J. {UCS}*, 17(2):276–295, 2011.

[FBB13] Mario Frank, Joachim M Buhman, and David Basin. Role mining with probabilistic models. *ACM Trans Inf Syst Secur*, 15(4):15, 2013.

[FHHM14] Johann Füller, Katja Hutter, Julia Hautz, and Kurt Matzler. User Roles and Contributions in Innovation-Contest Communities. *J. Manag. Inf. Syst.*, 31(1):273–308, 2014.

[FMP11] Fabian Friedrich, Jan Mendling, and Frank Puhlmann. Process Model Generation from Natural Language Text. In Haralambos Mouratidis and Colette Rolland, editors, *Adv. Inf. Syst. Eng. - 23rd Int. Conf. CAiSE 2011, London, UK, June 20-24, 2011. Proc.*, volume 6741 of *Lecture Notes in Computer Science*, pages 482–496. Springer, 2011.

[GKS08] Georgios Gousios, Eirini Kalliamvakou, and Diomidis Spinellis. Measuring developer contribution from software repository data. In *Proc. 2008 Int. Work. Conf. Min. Softw. Repos.*, pages 129–132. ACM, 2008.

[GPvD14] Georgios Gousios, Martin Pinzger, and Arie van Deursen. An exploratory study of the pull-based software development model. In *Proc. 36th Int. Conf. Softw. Eng.*, pages 345–355. ACM, 2014.

[Gre06] Shirley Gregor. The nature of theory in information systems. *MIS Q.*, pages 611–642, 2006.

[Hen17] Fergus Henderson. Software engineering at Google. *arXiv Prepr. arXiv1702.01715*, 2017.

[HMCX14] Qiuju Hou, Yutao Ma, Jianxun Chen, and Youwei Xu. An Empirical Study on Inter-Commit Times in {SVN}. In Marek Reformat, editor, *26th Int. Conf. Softw. Eng. Knowl. Eng. Hyatt Regency, Vancouver, BC, Canada, July 1-3, 2013.*, pages 132–137. Knowledge Systems Institute Graduate School, 2014.

[HMPR04] Alan R Hevner, Salvatore T March, Jinsoo Park, and Sudha Ram. Design Science in Information Systems Research. *MIS Q.*, 28(1):75–105, 2004.

[JM14] Dan Jurafsky and James H Martin. *Speech and language processing*. Pearson, 2014.

[KRS06a] Ekkart Kindler, Vladimir Rubin, and Wilhelm Schäfer. Activity Mining for Discovering Software Process Models. *Softw. Eng.*, 79:175–180, 2006.

[KRS06b] Ekkart Kindler, Vladimir Rubin, and Wilhelm Schäfer. Incremental Workflow Mining Based on Document Versioning Information. In Mingshu Li, Barry Boehm, and LeonJ. Osterweil, editors, *Unifying Softw. Process Spectr.*, volume 3840 of *Lecture Notes in Computer Science*, pages 287–301. Springer Berlin Heidelberg, 2006.

[KVSvdB12] Erik Kouters, Bogdan Vasilescu, Alexander Serebrenik, and Mark G. J. van den Brand. Who's who in gnome: Using LSA to merge software repository identities. In *28th IEEE International Conference on Software Maintenance, ICSM 2012, Trento, Italy, September 23-28, 2012*, pages 592–595. IEEE Computer Society, 2012.

[LBGL16] Aron Lindberg, Nicholas Berente, James Eric Gaskin, and Kalle Lyytinen. Coordinating Interdependencies in Online Communities: A Study of an Open Source Software Project. *Inf. Syst. Res.*, 27(4):751–772, 2016.

[Leo13] Henrik Leopold. *Natural Language in Business Process Models - Theoretical Foundations, Techniques, and Applications*, volume 168 of *Lecture Notes in Business Information Processing*. Springer, 2013.

[LHY+15] Haibing Lu, Yuan Hong, Yanjiang Yang, Lian Duan, and Nazia Badar. Towards user-oriented RBAC model. *J. Comput. Secur.*, 23(1):107–129, 2015.

[LM14] Sherlock A. Licorish and Stephen G. MacDonell. Understanding the attitudes, knowledge sharing behaviors and task performance of core developers: A longitudinal study. *Inf. Softw. Technol.*, 56(12):1578–1596, 2014.

[MH10] Walid Maalej and Hans-Jörg Happel. Can Development Work Describe Itself? *7th IEEE Work. Conf. Min. Softw. Repos. (MSR 2010)*, pages 191–200, 2010.

[MLP14] Jan Mendling, Henrik Leopold, and Fabian Pittke. 25 Challenges of Semantic Process Modeling. *Int. J. Inf. Syst. Softw. Eng. Big Co.*, 1(1):78–94, 2014.

[MM16] Ann Majchrzak and Arvind Malhotra. Effect of Knowledge-Sharing Trajectories on Innovative Outcomes in Temporary Online Crowds. *Inf. Syst. Res.*, 27(4):685–703, nov 2016.

[MSVA16] Barsha Mitra, Shamik Sural, Jaideep Vaidya, and Vijayalakshmi Atluri. A Survey of Role Mining. *ACM Comput Surv*, 48(4):1–37, 2016.

[OSGdS11] Gustavo A. Oliva, Francisco W.S. Santana, Marco A. Gerosa, and Cleidson R.B. de Souza. Towards a classification of logical dependencies origins. *Proc. 12th Int. Work. 7th Annu. ERCIM Work. Princ. Softw. Evol. Softw. Evol. - IWPSE-EVOL '11*, page 31, 2011.

[PSvdB11] Wouter Poncin, Alexander Serebrenik, and Mark van den Brand. Process Mining Software Repositories. *2011 15th Eur. Conf. Softw. Maint. Reengineering*, pages 5–14, 2011.

[PTRC08] Ken Peffers, Tuure Tuunanen, Marcus Rothenberger, and Samir Chatterjee. A Design Science Research Methodology for Information Systems Research. *J. Manag. Inf. Syst.*, 24(January):45–77, 2008.

[RdBS17] Pedro H Piccoli Richetti, João Carlos de A. R. Gonçalves, Fernanda Araujo Baião, and Flávia Maria Santoro. Analysis of Knowledge-Intensive Processes Focused on the Communication Perspective. In Josep Carmona, Gregor Engels, and Akhil Kumar, editors, *Bus. Process Manag. - 15th Int. Conf. {BPM} 2017, Barcelona, Spain, Sept. 10-15, 2017, Proc.*, volume 10445 of *Lecture Notes in Computer Science*, pages 269–285. Springer, 2017.

[RGV+07] Vladimir Rubin, Christian W Günther, Wil M P Van Der Aalst, Ekkart Kindler, Boudewijn F Van Dongen, and Wilhelm Schäfer. Process mining framework for software processes. In *Softw. Process Dyn. Agil.*, pages 169–181. Springer, 2007.

[RHL15] Jukka Ruohonen, Sami Hyrynsalmi, and Ville Leppänen. Time series trends in software evolution. *J. Softw. Evol. Process*, 27(12):990–1015, nov 2015.

[RLvdA14] Vladimir Rubin, Irina Lomazova, and Wil M P van der Aalst. Agile development with software process mining. In *Proc. 2014 Int. Conf. Softw. Syst. Process*, pages 70–74. ACM, 2014.

[RvdA06] Anne Rozinat and Wil M P van der Aalst. Decision mining in ProM. In *Int. Conf. Bus. Process Manag.*, pages 420–425. Springer, 2006.

[SCJM15] Stefan Schönig, Cristina Cabanillas, Stefan Jablonski, and Jan Mendling. Mining the Organisational Perspective in Agile Business Processes. In *BPMDS*, page In press., 2015.

[SvdA07] Minseok Song and Wil M P van der Aalst. Supporting process mining by showing events at a glance. In *Proc. 17th Annu. Work. Inf. Technol. Syst.*, pages 139–145, 2007.

[SvdA08] Minseok Song and Wil van der Aalst. Towards comprehensive support for organizational mining. *Decis. Support Syst.*, 2008.

[THB14] Stephen W Thomas, Ahmed E Hassan, and Dorothea Blostein. Mining unstructured software repositories. In *Evol. Softw. Syst.*, pages 139–162. Springer, 2014.

[vdA16] Wil M P van der Aalst. *Process mining: data science in action*. Springer, 2016.

[vDdA05] Boudewijn F van Dongen and Wil M P der Aalst. A Meta Model for Process Mining Data. *EMOI-INTEROP*, 160:30, 2005.

[vDdMV+05] Boudewijn F van Dongen, Ana Karla A de Medeiros, H M W Verbeek, AJMM Weijters, and Wil M P Van Der Aalst. The ProM framework: A new era in process mining tool support. In *Appl. Theory Petri Nets 2005*, pages 444–454. Springer, 2005.

[VPHB16] John Venable, Jan Pries-Heje, and Richard Baskerville. FEDS: a Framework for Evaluation in Design Science Research. *Eur. J. Inf. Syst.*, 25(1):77–89, jan 2016.

[WB18] Bartosz Woliński and Saimir Bala. Comprehensive Business Process Management at Siemens: Implementing Business Process Excellence. In Jan vom Brocke and Jan Mendling, editors, *Business Process Management Cases: Digital Innovation and Business Transformation in Practice*, pages 111–124. Springer International Publishing, Cham, 2018.

[WBMT99] Ian H Witten, Zane Bray, Malika Mahoui, and Bill Teahan. Text Mining: A New Frontier for Lossless Compression. In *Proc. Conf. Data Compression*, DCC '99, pages 198—-, Washington, DC, USA, 1999. IEEE Computer Society.

[Wit04] Ian H Witten. Text Mining., 2004.

[YR07] Liguo Yu and Srini Ramaswamy. Mining CVS repositories to understand open-source project developer roles. In *Proc. - ICSE 2007 Work. Fourth Int. Work. Min. Softw. Repos. MSR 2007*, pages 7–10, 2007.

[YR14] Annie T T Ying and Martin P. Robillard. Developer profiles for recommendation systems. *Recomm. Syst. Softw. Eng.*, pages 199–222, 2014.

[YSX13] Weicheng Yang, Beijun Shen, and Ben Xu. Mining GitHub: Why Commit Stops - Exploring the Relationship between Developer's Commit Pattern and File Version Evolution. In Pornsiri Muenchaisri and Gregg Rothermel, editors, *20th Asia-Pacific Softw. Eng. Conf. {APSEC} 2013, Ratchathewi, Bangkok, Thailand, December 2-5, 2013 - Vol. 2*, pages 165–169. {IEEE} Computer Society, 2013.

[ZN08] Thomas Zimmermann and Nachiappan Nagappan. Predicting defects using network analysis on dependency graphs. *Proc. 13th Int. Conf. Softw. Eng. - ICSE '08*, page 531, 2008.

[ZRDvD08] Andy Zaidman, Bart Van Rompaey, Serge Demeyer, and Arie van Deursen. Mining Software Repositories to Study Co-Evolution of Production & Test Code. In *ICST*, pages 220–229. {IEEE} Computer Society, 2008.