PhD Research Plan

Agile Software Development Process Mining:
Discovery, Conformance Checking and Enhancement

João Carlos Palmela Pinheiro Caldeira
(Mestre)

PhD Program in Information Science and Technology

Supervised by Fernando Brito e Abreu, Associate Professor, DCTI, ISCTE-IUL
Co-supervised by Jorge Cardoso, Associate Professor, FCT/UC

Lisbon
July, 2015
Abstract

Background
Software development has become a fundamental process on any business or organization. As a consequence, together with other emergent technologies, new development platforms are being created, mainly in the cloud (e.g., Eclipse Orion, Cloud9, Codio, etc), requiring different approaches on the way software development can be studied. Traditionally, software evolution studies use data insights from software configuration management repositories (SCM), source code systems (SC) and bug tracking (BT) tools. Many times those works focus on SCM, SC and BT synergies but they do not include metadata collected from the development environments (IDE’s) about user activities. Information about developer behavior and IDE usage can be identified as a missing perspective and a fundamental dimension to be part in the area of agile software development.

Research problems
Due to the lack of IDE information in software related studies we have identified the major problems which we expect to mitigate: (RP1) Reduced agility in creating software process models from empirical studies mainly with a multi-dimension approach including user behavior metadata, (RP2) Difficult validation of developers adherence/compliance to agile methodologies, (RP3) Restricted ability to improve a development process and corresponding tools when not considering the human (behavioral) factor and lack of individual behavior insights when choosing the right/best developers to specific technical requirements.

Expected contributions
We expect to produce contributions across three areas of software process mining: model discovery, conformance checking and process enhancement. Model discovery aims at discovering a software process model or at least some software process patterns by mining event logs taken from real software development activities. Conformance checking stands for diagnosing if actual software development activities (again captured as event logs) are following a pre-specified model. Finally, process enhancement aims at improving an existing software process model with information extracted from actual software process instances, captured from event logs during the various activities of the software development process.

Based on the mentioned mining techniques, we expect to analyze the order of activities in development, highlight resources, such as people, projects, roles, and how they are related and potentially predict process time, discover bottlenecks, track resource utilization and measure service levels. From a Socio-Technical perspective we expect to identify bad practices and good practices amongst the developers and profile them using clustering analysis or other classification techniques. As an overall result we expect to contribute to make IDE’s more adaptable to the user by identifying different developer profiles, detecting the most used features and actions done in IDEs and potentially by suggesting advices to the developers.

Research methodology
This work is supported by a software artifact that has to be built in order to collect the desired data and will be used by a group of students having programming classes. Therefore, the adoption of the Design Science Research methodology is fundamental in validating the prototype that supports our major tasks. The methodology for those tasks comprises also the usage of the process mining guiding principles and their three derivatives: discovery, conformance checking and enhancement to tackle the identified research problems. Along our study we may also find the need to use the scientific method to validate some results and accept or reject hypothesis.

Keywords: software repositories, software development IDE’s, process mining, big data technologies
1. Context

As a result of the direct or indirect dependence on software in our daily lives, software development has become a fundamental process on any business or organization. Consequently, it is vital to carefully study, understand and improve such a process [1]. However, the development process is often not formalized (i.e. a model is not available), especially in agile methodologies such as Scrum, Extreme programming (XP), Feature-driven development (FDD), Agile unified process (AUP) or Open Unified Process (OpenUP).

While agile approaches give a wide berth for a development team to define its own development process, a set of stereotypical activities should be in place. For instance, in XP, developers are expected to record user stories, write tests before coding, program in pairs, use refactoring techniques, do check-ins frequently (continuous integration), adopt a coding standard, own the code collectively (i.e. anyone can improve any part of the code at any time) and do not work beyond 40 hours a week [2]. The efficiency and effectiveness of agile approaches relies on the adoption of those activities, but due to the invisibility of software development activities [3], it is not straightforward to check if members of a development team are actually performing them or not.

Regarding the quality of the software development process, several approaches have been proposed to improve and assess it, either at the organizational level (e.g. CMMI - Capability Maturity Model Integration [4]), project level (e.g. Team Software Process [5]) or individual level (e.g. Personal Software Process [6]). Those approaches were conceived for non-agile software development processes in mind and rely heavily on collecting evidence of current practice. This often implies a considerable overhead of manual data collection regarding software process activities. Along this intrusiveness, collected data is often adulterated due to the well-known Hawthorne Effect [7], therefore not allowing to reach valid conclusions. It is therefore not surprising that little progress has been made on researching the quality of agile software development [8].

Meanwhile, the increasing role of the open-source movement has allowed a considerable increase in the availability of free software engineering tools, namely in Eclipse, the dominant IDE for the dominant programming language, Java1. That availability has led to a progressive use of a multitude of tools during a software development project, far beyond the traditional ones that dominated the first decades of computer programming, namely the code editor, compiler, linker and configuration management system (CMS). Nowadays developers use modeling tools, code generators, code recommendation tools, code smell detection tools, refactoring tools, metrics collection tools, software structure visualization tools, test generators and test coverage tools, to name just a few. Those tools are becoming increasingly intertwined within the IDE and the latter is progressively integrated with cloud-based services that allow cooperative work (e.g. GitHub, Sourceforge) that provide services such as CMS, issue tracking system, project documentation and project wiki.

A recent survey mentions that around 2/3 of the data sources used for data analysis on software development projects come from CMS (e.g. CVS, SVN, GIT or Mercurial) [9]. These systems maintain a history of changes in all files, and control the authors responsible for the modifications. CMS data include information such as commit messages, commit time, commit author, and commit type (added, modified, or deleted). However, in agile software development

initiatives, using a multitude of techniques and tools as aforementioned, it is not possible to characterize and understand the process by just looking at the check-in and check-out events in a CMS. A more holistic approach is called for.

Last, but not the least, we are witnessing the birth of cloud-based IDEs. Examples include Eclipse’s Orion², Cloud9³ and Codenvy⁴. Albeit the obvious convenience of having the development environment available from any web-enabled device, little is known on the impact that such cloud-based IDEs will have on software development processes.

This document describes a PhD thesis proposal that is expected to provide agile software developers with a better awareness of their development process, thus allowing to align and improve it. In the next section we summarize the main research problems that will be addressed. In section 3 we will sketch the proposed solution to address those problems. Then, in section 4, some related work is surveyed. Finally, some methodological aspects related to our proposed work are briefly reviewed in section 5.

2. Research problems and hypotheses

In this section we identify some of the research problems where we plan to conduct our work. For each problem we drill down to specific research hypotheses, which are stated in their usual null formulation. Our research will then be targeted to provide evidence that will allow us to reject those null hypotheses.

2.1 (RP1) Process discovery in agile software development

Agile software development is usually performed by small teams, in short iterations. According to the famous Manifesto for Agile Software Development, agile teams are supposed to “value individuals and interactions over processes and tools” [10]. We believe this kind of strong claim was stated to contrast with highly structured / organized software development methodologies. By then, the most representative surrogate of the latter was the Rational Unified Process (RUP) that prescribed a proprietary set of practices organized in processes, and tools to support the same practices [11]. Agile approaches, like the aforementioned ones, also prescribe a set of agile practices. However, the way they are applied is left for each team to decide. Many of those practices (e.g. refactoring, regression testing) are unlikely to be applied without tools nowadays, so the cited manifesto claim became, at least partially, anachronic. Nevertheless, the process still remains mostly hidden from outsiders. Personal turnover in agile teams becomes a hindrance and things get worst if teams are geographically distributed, due to the tacit knowledge problem [12].

We claim that it is possible to discover the process of an agile team using a given IDE where a set of development tools are integrated, by mining the plethora of events generated by those tools. Our objective is that that process should be made available in a given modeling language (e.g. Petri Nets or BPMN) for the development team as a whole, showing where each team member is currently positioned.

The process discovery problem raises the following null hypotheses:

² https://orionhub.org/
³ https://c9.io/
⁴ https://codenvy.com/
[H$_{0,1a}$] Software development processes discovered from events recorded from tools used are insufficiently detailed to be useful for software engineering purposes.

[H$_{0,1b}$] It is not possible to provide feedback to agile developers in real time regarding the process being executed, namely regarding the current positioning of each team member.

2.2 (RP2) Compliance checking in agile software development

In the last few years agile development methodologies have gained major focus from the organizations and practitioners in general. Together with this, the development process has evolved from an individual task (e.g. using Eclipse) to a more collaborative one. This is supported by new tools delivered by public or private cloud IDEs and has introduced new challenges in validating the adherence of individuals to the methodologies (e.g., Scrum or XP) used by each organization or department. Conformance validation is a fundamental activity to implement rules and improve processes within any organization [13]. If each developer in a team is left alone, without a perception of his/her alignment with the expected process, ultimately it may lead to a lack of quality in the delivered products.

The process compliance checking problem raises the following null hypotheses:

[H$_{0,2a}$] There is no significant distinction between what agile developers do in practice and what they were supposed to do (as prescribed in a model).

[H$_{0,2b}$] There is no significant variability in the roles performed within a development team in agile approaches.

2.3 (RP3) Process enhancement in agile software development

Every project has its own needs in terms of requirements, technologies and human resources that should be allocated to each task. The software development process not only needs the right people with the right skills in programming but also the right people in terms of behavior. Productivity is derived from both: skills and behavior. Software development is in fact a socio-technical activity [1] and to improve software we need to take into consideration these two elements. A major difficulty in identifying the best resources for a specific project is caused by the fact that we don’t know with enough accuracy how programmers behave using cloud IDEs individually or in groups when they are coding software. Having this information available can add a useful new dimension on software engineering related studies and contribute to the software improvement.

A major contribution to a development project or any process in general is the ability to evaluate the quality of the processes and tools used in the programming and to identify ways to improve them both. Difficulties in researching on software development processes and tools improvement have been highlighted by the industry [14] [15] and in the literature [16]. These difficulties have in common a lack of methods to mine the processes and artifacts usage. With a very limited understanding of a process or a tool usage, trying to improve them can be very inconsequent tasks. Following this, the main questions we can put currently are: Can we extract relevant information from the IDE in order to make it adaptable to end user profile? Can we really use it to improve the overall process?
The process enhancement problem raises the following null hypotheses:

[H₃a] It is not possible to provide feedback to the agile developer regarding the quality of the process he/she is executing.

[H₃b] There is not a significant improvement in software process quality due to the use of cloud-based IDEs.

3. Proposed solution

3.1 Introduction

To foster a shared understanding of what the current process really is, we will use model discovery techniques. The latter allow to reverse engineer the software process model by mining event logs taken from real software development activities. Those activities are expected to characterize the underlying process model, since a given execution flow (sequence of consecutive or parallelized executed activities, from start to end, corresponds to what we call a “process instance”.

We will use conformance checking techniques for diagnosing if actual software development activities (again captured as event logs) are following a given process model. Our objective here will be to provide each developer with an “agile dashboard” where the current adoption of the agile activities will be gauged against best practices. The latter can be based on values taken from the best results obtained with the team or the organization.

Finally, process enhancement aims at improving an existing software process model with information extracted from actual software process instances, once again captured as logs of events raised during the various activities of the software development process. We expect to devise and highlight the most frequent activity paths in development, highlight resources, such as people, systems, roles, and how they are related and potentially predict process time, discover bottlenecks, monitor resource utilization and, measure service levels. From a socio-technical perspective, we expect to identify bad practices and good practices amongst the developers and profile them using clustering analysis or other classification techniques. Another issue that we expect to address with our software process mining based approach is identifying the friction factors that have a negative impact on the software development pace.

In this PhD research work we expect to adopt a holistic approach where events generated by all tools used will be considered. A standardized format represented in Figure 1 for recording events is will be used for the sake of interoperability, namely with process mining tools.

The number of events to be generated by a development team within a project iteration (e.g. a Scrum sprint) can be very large. We will therefore assess if a big data open-source platform (e.g., Cassandra or HBase) will be required for storing and processing the collected data in our research.

---

5 http://www.xes-standard.org/
6 http://cassandra.apache.org/
7 http://hbase.apache.org/
The aforementioned approach is expected to scaffold exploratory activities on top of the collected data, allowing the community to do benchmarking, evaluate software engineering best practices and assess software engineering research topics like the ones identified in 2 by means of structured empirical studies [17]. We firmly believe that a combination of process mining techniques using machine learning algorithms supported by big data technologies would be the best approach to tackle the identified research problems [18] [19] [20]. Due to the fact that we might be dealing with large amounts of data with different formats, coming from different sources, the techniques and tools to perform software analysis should be aligned with the challenges imposed by those scenarios. As pointed earlier by [21] big data technologies not only deal with the data challenges mentioned above but also in leveraging visualization capabilities as explained later in the next section.

3.2 Proposed solutions (PS)

The following subsections present the proposed solutions for each of the problems identified in section 2. The Proposed Solutions are presented with the initials PS and number that identify the problem.

3.2.1 PS1 – Orion IDE Plugin development and initial process mining

To have a standard and automated process to collect data can mitigate the efforts to validate results and sustain conclusions. Recently [22], was developed an Eclipse plugin to extract metrics using a meta-model driven measurement approach. This method brings agility and flexibility to the process of collecting metrics data from Eclipse IDE. We plan to take one step further and try to port the same principles to the Eclipse Orion cloud IDE. Apart from the plugin to collect the events from the IDE functions, the fundamental activity is to mine the data by using one type of process mining – discovery.
Validation experiences are being prepared within the context of programming classes such as “Web Application Development” - [https://fenix.iscte.pt/disciplinas/00699/](https://fenix.iscte.pt/disciplinas/00699/), under the responsibility of Professor Carlos Costa from the CTI department in ISCTE, and also with Universidade Europeia, under the responsibility of Master Américo Rio.

The goal is to use the Orion IDE and the plugin as a beta environment in their programming classes so we can test the collection mechanisms and refine the granularity of the events obtained. In addition, once the plugin is developed and if we wish to make it available in the cloud, anyone using public or private Orion IDE’s can use it. This method will allow us to immediately discover software development process models.

In addition, during programming classes of Software Engineering I and II of Prof. Fernando Brito e Abreu, we will assess if we can also gather event log data from the usage of MyLyn\(^8\), a task and application lifecycle management tool which monitors developers programming activities in order to create a "task context" that automatically links all relevant artifacts to the task-at-hand. This puts the information developers need at their disposal and improves productivity by reducing information overload, facilitating multitasking and easing the sharing of expertise.

### 3.2.2 PS2 – Process adherence conformance checking

To understand developers adherence to some agile methodologies such as Scrum and others, we intend to use - conformance checking [23].

Using this approach we will cover different perspectives such as: i) control-flow perspective, responsible for analyzing the order of activities, ii) organizational perspective, which focuses on highlight resources, such as people, systems, roles and how they are related, iii) case perspective, characterized by the actors working on a process or its own path, and iv) the time perspective concerned with frequency of events and their timing, allowing us to predict remaining process time, discover bottlenecks, monitor resource utilization and measure service levels.

### 3.2.3 PS4 – Process development and tools improvement

Process improvement and mainly software development improvement are some of the most active topics in the research community and software industry [10].

Using mainly the organizational and case perspectives we expect to identify unknown relations between actors, patterns in programming, eventually detect trends and perform predictions and therefore understand better the developer’s behavior. Clustering and classification are techniques we might use in order to build developers profile using the two mentioned dimensions: skills and behavior. Following this, unveiling bad practices and good practices is relevant in defining roles and allocating the right resources to satisfy specific project requirements in the future. Having the right people doing the right tasks contribute to improve the productivity and the overall quality of the software.

IDE events log data can be a valuable data source to improve the overall process. Improving the process means we may have to look also to improve the tool being used. Therefore, understanding how the IDE is used and making sure that it is adaptable to each developer role (e.g. by suggesting plugins that peers with similar behavior are using) and characteristics would increase also the

\(^8\) [http://www.eclipse.org/mylyn/](http://www.eclipse.org/mylyn/)
productivity by improving the coding tasks. This approach is being recommended by the industry and European organizations for future software engineering related studies [15].

Nevertheless, this is yet to be proven, and we expect to validate it together with the users and answer to the questions in RP3.

4. Related work

4.1 Process Mining

Process mining techniques gives us the possibility of extracting knowledge from a variety of information systems event logs. They provide new means to discover, monitor and improve processes in several domain areas [24]. Typically, these event logs can be used to discover roles in organizations (e.g., groups of people with similar work patterns) which can be used to relate individuals and their particular activities.

There are three basic types of process mining: i) discovery, ii) conformance checking, and iii) enhancement.

![Three basic types of process mining](image)

Figure 2 - Three basic types of process mining in terms of input and output (adapted from [10])

Several techniques for process mining have been used, including generalization, characterization, classification, clustering, association, evolution, pattern matching, data visualization, and meta-rule guided mining to contribute to software analysis [25].

This thesis is related with the process of mining software development activities, the application of big data technologies and visualization methods to contribute to understand and improve the software development processes. In the following sections we will describe them briefly.

4.2 Agile Methodologies

The term Agile in a software development context was first introduced in 2001 by a group of software developers [26]. From this work derived the main principles of modern development methodologies. From the main guiding values of the manifesto we can highlight as the focus of the methodology the following: i) Individuals and interactions over processes and tools, ii) Responding to change over following a plan to deliver results to the stakeholders.

Literature has been produced about some performed studies related with agile methodologies [27], [16], [13], [23]. These studies were performed on software being used by the end-users and with the focus on the functionalities it should deliver. What they have in common is that they lack...
the developer behavior perspective during the programming phase which would address the above described guiding values.

Agile methodologies (e.g., Scrum, Extreme programming (XP), Feature-driven development (FDD), Agile unified process (AUP), etc.) are becoming the de-facto standard and validating process compliance is one of the key challenges within this research topic. Process mining can be of great value in discover process models and also in the conformance checking to validate the adherence to agile methodologies. At the same time, process development improvement and IDE adaptability can emerge as the major outcomes of mining data in real-time directly from the IDE.

4.3 Big Data

The term Big Data is currently extremely popular within the IT market [28]. It is also becoming a hot research topic [21], [29]. It refers to the ability of collecting, storing, analyzing and visualizing large amounts of data. Usually this data is so large and sometimes so complex that traditional tools and data processing applications are no longer able to process it within a tolerable time. It is common to accept a system as a candidate for a big data use case when it falls under these three basic dimensions (3Vs): volume, variety and velocity. They correspond to the amount of data being captured, the number of different types of data and to the speed it has to be processed in order to provide relevant and timely results to the stakeholders. Usually this data is a relevant factor when to perform analytical functions within a specific area of business [19], [20].

![Figure 3 - High Level architecture for the collection, archiving and querying solution](image)

We plan to use big data technologies, as viewed in Figure 3, for collecting data and as a repository for our research artifacts [30]. It does a perfect match to address the requirements spawned by software evolution visualization and also provides the scalability mechanisms that the collection of large volumes of data might require in a short to medium term.

5. Research methodologies

This research endeavor will apply a combination of two methodologies: the Scientific Method (SM) and the Design Science Research (DSR). The latter will be used in the conception and operationalization of the research instrument described in section 3. This instrument will allow collecting data that will be used for assessing our research hypotheses, as prescribed by the SM.
5.1 SM approach

While delving into the research problems that were previously identified (section 2), we formulated several research hypotheses. The SM is a fundamental technique used by scientists to raise hypothesis and produce theories. A theory is a conceptual framework that explains existing facts or predicts new facts. It assumes that the scientific knowledge is predictive and that cause and effect relationships exist. Knowledge in an area is expressed as a set of theories and theories are raised upon non refuted hypothesis. The SM progresses through a series of steps: (i) observe facts, (ii) formulate hypotheses, (ii) design and (iii) execute de experiment (implies the availability of collection instruments and subjects from where data can be collected), (iv) analyze data and interpret the results, (v) raise a theory and, (vi) disseminate results for peer validation.

5.2 DSR approach

It has been reported in the literature a shortage of empirical studies in the information technology domain and more specifically in software development [1], [9], [3]. This is often related with the lack of consistent methods in data gathering or lack of real life use cases. As a result, the data analysis is not trustworthy, and the findings of those studies cannot easily be validated within the research community. We expect that our automated data collection approach will mitigate this problem. As such we will be able to test the set of hypothesis that will emerge from the research problems that were described in section 2.

DSR has its roots in engineering and is appropriate when developing new technologies for solving problems, such as the ones described herein. DSR helps gaining problem understanding, identifying systemically appropriate solutions, and in effectively evaluating new and innovative solutions. The DSR methodology prescribes several activities that are being adapted to our context: (i) problem identification and motivation section, (ii) definition of the objectives for a solution, (iii) design and development, (iv) demonstration, (v) evaluation and, (vi) communication / dissemination.

Both SM and DSR approaches encompass the publication of results for peer scrutiny. We will privilege narrow-scope conferences and journals with a high rank, to better focus on our research concerns, and also because these are the forums where the best researchers of the relevant community are expected to present their works. Submitting to those forums will enable us to maximize the quality of the received feedback from our work, even in rejection situations. The ones that we have identified so far include:

- International Conference on Software and Systems Process
- Journal of Software: Evolution and Process

Bibliography


