Software Development Process Mining: Discovery, Conformance Checking and Enhancement

Conference Paper · September 2016

DOI: 10.1109/QUATIC.2016.061

2 authors:

João Caldeira
Universidade Lusófona
8 PUBLICATIONS **70** CITATIONS
SEE PROFILE

Fernando Brito e Abreu
ISCTE-Instituto Universitário de Lisboa
218 PUBLICATIONS **2,442** CITATIONS
SEE PROFILE

Software Development Process Mining:

Discovery, Conformance Checking and Enhancement

João Caldeira
Fernando Brito e Abreu
Instituto Universitário de Lisboa (ISCTE-IUL)
ISTAR-IUL
Lisboa, Portugal
{jcppc, fba}@iscte-iul.pt

***Abstract*—Software development has become a fundamental process on any business or organization. As a consequence, together with other emergent technologies, new development platforms (IDEs) are being created, mainly in the cloud (e.g., Eclipse Orion, Cloud9, Codio), requiring different approaches on the way software development can be studied. Empirical studies on software development most often are based on data taken from software configuration management repositories, source code management systems and issue tracking tools, but not from the IDEs themselves, because they do not record data publically regarding developers' activities. We aim to bring forward new insights on the software development process by analyzing how developers use their IDE. Based upon process mining techniques such as process discovery and conformance checking, this missing perspective will hopefully allow the discovery of coding patterns, the search for programmer behaviors and the detection of deviations from prescribed processes. Finally, we expect to provide advice for individual software process enhancement.**

Keywords—software development, software process, process mining, pattern discovery, software engineering

I. Introduction

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 frequently implies a considerable overhead of manual data collection regarding software process activities. Along this intrusiveness, collected data is regularly 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 its most widespread programing language, Java. 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 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, such as *Eclipse Che*[1], *Orion*[2], *Cloud9*[3] and *Codenvy*[4]. Albeit the obvious convenience of having the development environment available on a web browser on any web-enabled device, little is known on the impact that cloud IDEs will have on software development processes.

This document describes a research initiative that is expected to provide software developers with a better awareness of their development process, thus unveiling enhancement opportunities, by mining IDEs event logs. In section II we address some related work studies, and in section III we summarize the main research problems that will be addressed and some methodological aspects are briefly reviewed. The solution proposed is presented in section IV. Finally, section V draws some conclusions and makes provisions for future work.

II. Related Work

Process mining techniques give us the possibility of extracting knowledge from a variety of information sources, especially from event logs. They provide new means to discover, monitor and improve processes in several domain areas [10]. Typically, these event logs can be used to discover roles in organizations (e.g., groups of people with similar work patterns).

There are three basic types of process mining: i) discovery, ii) conformance checking, and iii) enhancement and we expect to leverage them in our research. These types of mining have been used by [11] to propose rules to merge event logs for generic process mining. An analysis of model precision and decomposition is presented in [12] thus justifying the current adoption of process mining in multiple domains. [13]–[15] use it in the context of mining software repositories, whilst we expect to use it in mining IDE usage and developer behavior. A research on several mining techniques potentially applicable to the software engineering process as a whole is addressed by [16], thus corroborating our approach. User behavior is the focus of [17] and it addresses how the end-user uses the software, but not how the developer has developed it. As a confirmation of using process mining to extract knowledge from a process, we observe in [18] that even from a service perspective this approach is a valid option.

The term Agile in a software development context was first introduced in 2001 by a group of software developers [10]. From this work derived the main principles of modern development methodologies and literature has been produced about some performed studies related with it [13], [16], [17], [19]. These studies were performed on software being used by the end-users and with the focus on the functionalities it should deliver. However, they lack the developer behavior perspective during the programming phase, therefore not providing evidence regarding the adoption of agile development practices and values.

Agile methodologies [20] are becoming the *de facto* standard and validating process conformance is one of the key challenges within this research thread. Process mining can be of great value in discovering process models and also in conformance checking, to validate the adherence to agile methodologies. Furthermore, we expect that process development improvement opportunities and IDE adaptability advices can emerge as major outcomes of mining data in real-time directly from the IDE.

In the research work presented in [21], the goal is to monitor and evaluate the maturity of agile methodologies but using other methods and data sources than the ones we expect to use. This proves the need for alternative and yet complementary ways of assessing these methodologies. A quantitative and qualitative approach is followed in [22] with the goal of analyzing the relationships among two dimensions of software development agility: response extensiveness and response efficiency. Nevertheless, this study is based on surveys and not sustained by real time automated data collection. Mining techniques have already been used in the scope of the software development process, but for a different objective and using different data sources: analyzing problems occurring during that process, based on data from an issue-tracking system [23].

Big data and machine learning techniques are extremely popular within the IT market [24] and their application on different problems also became a hot research topic [25]–[27]. They refer 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. Big data is often a critical success factor for performing analytical functions within a specific business area [19], [17] and our domain is no exception.

We plan to use big data technologies for collecting data, as a repository for our research artifacts [28] and machine learning as a layer to do analysis on it. It does a perfect match to address the requirements spawned by this research domain and also provides the scalability mechanisms that the collection of large volumes of data coming from IDE usage may require in a short to medium term. In a real world scenario, we may need to collect multiple events per second from each developer. Therefore, having processing speed, scalability and archiving efficiency is mandatory to support this research.

III. Research Objectives and Methodology

In this section we identify some of the research problems where we plan to focus our research. 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.

[1] www.eclipse.org/che

[2] planetorion.org

[3] c9.io

[4] codenvy.com

A. *Process Discovery in Software Development*

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, an anachronism. 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 making that process explicit, using an appropriate modeling language (e.g. Petri Nets or BPMN), either for a single developer or for a development team as a whole, evidencing the role of each team member. As an example, we may identify the activities a developer executes when using the IDE, such as: open projects, add contents, remove contents, refactoring, and save a project, among others.

The process discovery problem raises the following hypotheses:

- [H01a] Software development processes discovered from events recorded from development tools used are insufficiently detailed and accurate to be useful for software engineering purposes.
- [H01b] It is not possible to provide feedback to agile developers in real time regarding the process being executed, namely by being able to differentiate the role of each team member.

B. *Conformance Checking in Software Development*

In the last few years, agile development methodologies became mainstream in software development organizations. Meanwhile, the development process has evolved from an individual task to a more collaborative one. This is supported by new tools delivered by public or private IDEs and has introduced new challenges in validating the adherence of individuals to the agile methodologies supposedly adopted in place by each organization or department. Conformance validation can be used to check process 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 conformance checking problem raises the following hypotheses:

- [H02a] It is not possible to identify divergences between what agile developers do in practice and what they were supposed to do (as prescribed in a process model) by mining event logs generated by IDE usage.
- [H02b] There is no significant variability in the roles performed within a development team in agile approaches.

C. *Process Enhancement in Software Development*

Software development is a socio-technical activity [1]. Every project has its own needs in terms of requirements, technologies and human resources that should be allocated to each task. Successful software development projects not only require people with the right programming skills, but also with the right behavior. Productivity is derived from both: skills and behavior. A major difficulty in identifying the best human resources for a specific project is caused by the fact that we often have no clue on how programmers behave individually or in groups, while developing software. That behavioral information provides a new perspective that may contribute to improve software engineering project management and/or the individual adoption of best practices. Moreover, getting information on the installed plugins, their co-occurrence and their usage patterns may provide interesting insights that can be used to generate IDE configuration advices.

Difficulties in researching on software development processes and tools improvement have been reported by industry [14] [15] and academia [16]. These difficulties have in common a lack of methods to mine the processes and artifacts usage. Indeed, a limited awareness of actual processes and tools usage, hampers our ability to improve them. Following, the main questions we can put forward are: Can we extract relevant information from the IDE, in order to make it adaptable to the developer's profile? Can we really use it to improve the overall process?

The process enhancement problem raises the following hypotheses:

- [H03a] It is not possible to provide feedback to the agile developer, regarding the quality of the process he/she is executing.
- [H03b] It is not possible to automatically adapt the IDE to improve the developer's performance, based on the analysis of the IDE event logs.

D. *Methodology*

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 that will be described in section IV. This instrument will allow collecting data that will be used for assessing our research hypotheses, as prescribed by the SM.

While delving into the research problems that were previously identified, 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 the 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.

It has been reported in the literature a shortage of empirical studies in the information technology domain and more specifically in software development [1][3][9]. This is repeatedly 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, to be described in section IV, will mitigate this problem. As such, we will be able to test the aforementioned set of hypothesis that emerged from our research problems.

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.

IV. CURRENT WORK

To foster a shared understanding of what the current process really is, we will use model discovery techniques. It allows 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 research work we expect to adopt a holistic approach where events generated by all tools used will be considered. A standardized format represented by XES [10] events 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*[5] or *HBase*[6]) will be required for storing and processing the collected data in our research.

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 we have previously identified, by means of structured empirical studies [17]. Since we will be dealing with large amounts of data with different formats, coming from different sources, the techniques and tools to perform software analysis must 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 to foster qualitative perception and reasoning. 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].

A. *Eclipse Plugin Development and Initial Process Mining*

A standard and automated process to collect IDE data can mitigate the efforts to validate results and sustain conclusions. We plan to leverage Eclipse IDE events logged and potentially take one step further and try to port the same principles to a 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 – process discovery.

[5] - cassandra.apache.org

[6] - hbase.apache.org

We have built a preliminary version of an *Eclipse* events capture plugin that sends those events, wrapped as JSON objects, across a microservices architecture to a cloud server. The latter will convert the events to the XES format [10] and will apply process mining and machine learning algorithms. We are currently performing some beta tests to find out which is the adequate granularity level of the relevant events and filter out the remaining ones. Some preliminary validation experiments are being prepared within the context of programming classes in the context of several software development courses on two public universities in the Lisbon area *(ISCTE-IUL*[7] and *UNL*[8]*)*.

Once our experimental setup is validated, we plan to make the event-capture plugin available in the *Eclipse Marketplace*[9] for public download. Users of this plugin will be offered back a private dashboard on our cloud server, where they will be able to observe their profile and historical data on their own development process. Furthermore, they will get IDE configuration advices (e.g. suggestions on plugins used by peers with a similar profile). This approach will hopefully foster widespread participation and will allow us to conduct several wide scale experiments on software development process mining.

B. Process Adherence Conformance Checking

To understand developers adherence to some agile methodologies, such as *Scrum* and others, we intend to use - conformance checking techniques [23]. The latter cover different perspectives such as: i) control-flow perspective, responsible for analyzing the order of activities, ii) organizational perspective, which focuses on highlighting 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.

C. Development Process and Tools Improvement

Process enhancement 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 will improve our understanding on developer's behavior and expect to identify development process patterns, detect trends and perform predictions. We consider using clustering and other classification techniques to build developers profiles upon the two aforementioned dimensions: skills and behavior. Some profiles will be surrogates of good practices and others of bad practices. Albeit that profiling will mainly be for private gauging and improvement, like in the *Personal Software Process* approach [6], matching those profiles to the required roles in a development team may turn out to be a very useful tool in allocating the most adequate developers to satisfy specific project requirements. Having the right people doing the right tasks contributes to improve productivity and overall software quality.

[7] www.iscte-iul.pt/en

[8] www.unl.pt/en

[9] marketplace.eclipse.org

V. Conclusion

Most empirical studies on software-related topics cover product issues. As for the ones targeting the process dimension, many open research problems have been identified [1]. We observed in the literature that there is a lack of understanding on how developers behave during the development process itself, while using their main workbench – the IDE. Current IDEs are indeed toolboxes that offer a large plethora of facilities beyond the traditional edit-compile-run ones, such as debuggers, target runtime emulators, code generators, testbed tools, auditing, code visualization or lifecycle management tools. Those tools, offered as extensions to the IDE (aka plugins) generate events that can be trapped by the IDE. We are developing a cloud-based architecture that will analyze those events using process mining and classification techniques. By applying machine learning techniques on data collected from many developers, we expect to derive a set of profiles that will help characterizing developer roles along several perspectives.

Our approach is aligned with recent European recommendations for future software engineering related studies [15]. We expect to corroborate those recommendations by falsifying the null hypotheses stated in section III. If so, this new research thread will bring a new analytics dimension in the software development process engineering domain.

References

[1] A. Fuggetta, E. Di Nitto, and P. Milano, "Software Process," *Proc. Futur. Softw. Eng.*, pp. 1–12, 2014.

[2] K. Beck, *Extreme Programming Explained: Embrace Change*. 2004.

[3] F. P. J. Brooks, "No silver bullet-essence and accidents of software engineering," *Proc. IFIP Tenth World Comput. Conf.*, pp. 1069–1076, 1986.

[4] M. B. Chrissis, M. Konrad, and S. Shrum, *CMMI for Development: Guidelines for Process Integration and Product Improvement*. Pearson Education, 2011.

[5] W. S. Humphrey, *Introduction to the Team Software Process(sm)*. Addison-Wesley Professional, 2000.

[6] W. S. Humphrey, *Introduction to the Personal Software Process*. Addison-Wesley Professional, 1997.

[7] J. G. Adair, "The Hawthorne effect: A reconsideration of the methodological artifact," Journal of Applied Psychology, vol. 69, nº 2, pp. 334-345, May, 1984.

[8] I. Dubielewicz, B. Hnatkowska, Z. Huzar, and L. Tuzinkiewicz, "Quality Assurance in Agile Software Development," *Adv. Appl. Model. Eng.*, pp. 155–176, 2014.

[9] R. L. Novais, A. Torres, T. S. Mendes, M. Mendonça, and N. Zazworka, "Software evolution visualization: A systematic mapping study," *Inf. Softw. Technol.*, vol. 55, no. 11, pp. 1860–1883, 2013.

[10] W. Van Der Aalst, et al, "Process mining manifesto," *Lect. Notes Bus. Inf. Process.*, vol. 99 LNBIP, pp. 169–194, 2012.

[11] D. Teams, "Rational Unified Process Best Practices for Software," *Development*, pp. 1–21, 2004.

[12] A. CHUA and S. PAN, "Knowledge transfer and organizational learning in IS offshore sourcing," *Omega*, vol. 36, no. 2, pp. 267–281, 2008.

[13] V. Rubin, I. Lomazova, and W. M. P. van der Aalst, "Agile development with software process mining," *Proc. 2014 Int. Conf. Softw. Syst.*

Process - ICSSP 2014, pp. 70–74, 2014.

[14] N. W. Paper, “SOFTWARE Key Enabler for Innovation,” no. July, 2014.

[15] N. E. Software and S. Initiative, “Networked European Software and Services Initiative Complementary Recommendations for WP 2016 / 2017 on SOFTWARE ENGINEERING,” no. October, pp. 2014–2017, 2014.

[16] W. Poncin, A. Serebrenik, and M. Van Den Brand, “Process Mining Software Repositories,” *2011 15th Eur. Conf. Softw. Maint. Reengineering*, pp. 5–14, 2011.

[17] D. Zhang and T. Xie, “Software analytics in practice: mini tutorial,” p. 997, Jun. 2012.

[18] W. Van Der Aalst, and S. Member, “Service Mining : Using Process Mining to Discover, Check, and Improve Service Behavior” vol. 6, no. November, pp. 525–535, 2013.

[19] D. Zhang, Y. Dang, S. Han, and T. Xie, “Teaching and Training for Software Analytics,” in *2012 IEEE 25th Conference on Software Engineering Education and Training*, 2012, pp. 92–92, 2012.

[20] M. Brhel, H. Meth, A. Maedche, and K. Werder, “Exploring principles of user-centered agile software development: A literature review,” *Inf. Softw. Technol.*, vol. 61, pp. 163–181, 2015.

[21] R. M. Fontana, V. Meyer, S. Reinehr, and A. Malucelli, “Progressive Outcomes: A framework for maturing in agile software development,” *J. Syst. Softw.*, vol. 102, pp. 88–108, 2015.

[22] G. Lee and W. Xia, “Toward agile: An integrated analysis of quantitative and qualitative field data on software development agility,” *MIS Q.*, vol. 34, no. 1, pp. 87–114, 2010.

[23] R. Nayak and T. Qiu, “A Data Mining Application: Analysis of Problems Occurring During a Software Project Development Process,” *Int. J. Softw. Eng. Knowl. Eng. IJSEKE*, vol. 15, no. 4, pp. 647–663, 2005.

[24] Y. Simmhan, S. Aman, A. Kumbhare, R. Liu, S. Stevens, and Q. Zhou, “Cloud-based software platform for data-driven smart grid management,” *Comput. Sci. Eng.*, vol. 15, no. 4, pp. 1–11, 2013.

[25] R. Bryant, R. Katz, and E. Lazowska, “Big-Data Computing: Creating Revolutionary Breakthroughs in Commerce, Science and Society,” *Comput. Res. Assoc.*, pp. 1–15, 2008.

[26] G. Liu, M. Zhang, and F. Yan, “Large-Scale Social Network Analysis Based on MapReduce,” in *2010 International Conference on Computational Aspects of Social Networks*, 2010, pp. 487–490, 2010.

[27] N. Chen, S. C. H. Hoi, and X. Xiao, “Software process evaluation: a machine learning framework with application to defect management process,” *Empir. Softw. Eng.*, vol. 19, no. 6, pp. 1531–1564, 2014.

[28] A. Rajaraman and J. D. Ullman, “Mining of Massive Datasets,” *Lect. Notes Stanford CS345A Web Min.*, vol. 67, p. 328, 2011.