UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Software Engineering Curriculum

Mirlind Murati

Conformance Checking of Scrum Practices: A Study of 10 Open-Source Projects

Master's Thesis (30 ECTS)

Supervisors: Ezequiel Scott, PhD
Fredrik P. Milani, PhD

Tartu 2021

Conformance Checking of Scrum Practices: A Study of 10 Open-Source Projects

Abstract:
Software development teams add agility to their processes by implementing agile practices and frameworks, the correct execution of which, creates artifacts and traces in the development environments. This wealth of information can be used to compare the actual practices against the practices defined by the frameworks and, then, suggest potential improvements to the process. This thesis analyzed data taken from ten Jira repositories that belong to real life open-source projects. A set of Scrum rules is carefully extracted from official and relevant Scrum sources. In addition, several Scrum rules were proposed by the thesis. The goal of this thesis is to assess to what extent we can use data extracted from software development environments to verify the prescribed Scrum practices, through the aforementioned rules. Moreover, this thesis aims to apply rule-based conformance checking to the data, in order to verify the set of Scrum rules and check their compliance against the development team in a programmatic way. The results showed that the obtained data can only be mapped to nearly half of the defined Scrum practices and that, for the other half, there are not sufficient data captured by Jira. The results also indicated that some of the open-source projects are more compliant to the Scrum practices than the others, and that some Scrum practices themselves are more commonly adopted in the development teams than others. The thesis also highlights some limitations of the available data in issue-tracking software, which consequently conceal valuable information that can be used to improve team agility.

Keywords:
Scrum, Conformance Checking, Scrum Practices, Agile Software Development, Scrum Guide, Open-source

CERCS: P170 - Computer Science, numerical analysis, systems, control

Vastavuse kontroll Scrum'i alusel: 10 avatud lähtekoodiga tarkvaraprojekti uuring

Lühikokkuvõte:

Tarkvaraarendusmeeskonnad kasutavad oma protsessides agiilseid praktikaid ja raamistikke, mille nõuetekohane elluviimine loob arenduskeskkondadesse tehiseid ja jälgi. Suurt hulka sellest infost saab kasutada selleks, et võrrelda oma praktikaid praktikatega, mida defineerivad raamistikud, ja seejärel teha soovitusi oma praktikate täiustamiseks. Käesolevas magistritöös analüüsitakse andmeid, mis on pärit kümne avatud lähtekoodiga tarkvaraprojekti Jira repositooriumist. Lisaks defineeritakse ametlike ja asjassepuutuvate allikate alusel Scrum'i reeglite kogum ja pakutakse välja ideid uute Scrum'i reeglite kohta. Magistritöö eesmärk on selgitada välja, millises ulatuses on võimalik kasutada tarkvaraarenduskeskkondadest pärit andmeid, selleks et kaardistada vastavust Scrum'i reeglitele ja praktikatele. Lisaks teostab magistritöö andmetele reeglipõhist vastavuse kontrolli, et tuvastada Scrumi reeglite jälgimist arendusmeeskonnas. Magistritöö jõuti järeldusele, et andmeid saab seostada vaid umbes poolte defineeritud Scrum'i praktikatega. Ülejäänud praktikate kohta ei kogu Jira piisavalt andmeid. Magistritöö tuvastas ka tõsiasja, et mõned avatud lähtekoodiga projektid vastavad Scrum'i praktikatele paremini kui teised ja et arendusmeeskonnad jälgivad teatud Scrum'i praktikaid sagedamini kui teisi. Magistritöö toob esile ka mõningaid arenduskeskkondade piiranguid seoses andmete kättesaadavusega. Selle tulemusel jääb varjatuks väärtuslik informatsioon, mida muidu saaks kasutada meeskonna agiilsuse tõstmiseks.

Võtmesõnad:

Scrum, Vastavuse Kontroll, Scrum'i Praktikad, Agiilne Tarkvaraarendus, Scrum Guide, Avatud Lähtekoodigatarkvaraprojekti

CERCS:

P170 Arvutiteadus, arvanalüüs, süsteemid, kontroll

Contents

1 Introduction

The articulation of Agile Manifesto in 2001 [1], exactly two decades ago, has introduced revolutionary and unprecedented changes to the software engineering discipline. The appearance of agile methods is one of the most important evolutions for software development processes. They have now become the pivotal point of interest among development teams [2]. The software engineering realm has shifted from traditional development methodologies to agile development, in response to the dynamic user requirements and increasing software complexity [3]. Currently, the number of companies and software teams that embrace Agile practices is increasing. Software development based on these practices has proved to deliver improved productivity, satisfied clients and higher quality [4]. On the other hand, Business Process Mining is a novel field standing at the intersection of data mining and process modeling, and has shown successful application in several fields [5]. One of the capabilities that Process Mining encompasses, is Conformance Checking, which is used to inspect the consistency relation between a normative process model or a set of business rules against the actual behavior observed in an event log. Its goal is to examine and find similarities and deviations among the recorded behavior, i.e. the reality with the behavior of the prescribed model or the pre-defined rules [6].
As mentioned previously, many companies have been adopting agile methods in their development work during the past two decades, thus putting this methodology on an industrial pedestal. Moreover, there are numerous software process standards (e.g. ISO/IEC/IEEE 12207) and models (e.g. CMMI) which propose best practices for quality improvement of the process and the products that derive from those processes. On the other hand, some of the Agile methods (e.g. Scrum) are intentionally left incomplete [7], to enable teams for an implementation that more closely aligns to the company's culture, or to combine it with practices from other methods (e.g. TDD from XP). Nevertheless, such Agile implementations sometimes result in teams facing compliance and inconsistency issues between the agile working method they have adopted and the actual method as documented officially. Also, often times problems arise from having processes that deviate from the prescribed approach. According to the Scrum Guide, changing the core ideas of Scrum, omitting elements, or not following the rules of Scrum, limits the benefits deriving from Scrum, potentially even rendering it useless [8].
Since project management tools (e.g., Jira) are commonly used nowadays, they represent

a valuable source of data, the analysis of which can further our understanding on the reality of the (development) processes that have taken place in a certain project. We can utilize these data as event logs. This thesis focuses on the acquired data, or the event logs, which capture the reality of agile teams. It aims to find out whether these data, taken from collaborative software development environments, can be used to verify the implemented agile practices, by performing conformance checking with agile methods, which in this case is Scrum. I believe that valuable findings will enable discussions, provide insights to agile teams, and hopefully help them to improve their software development processes.

1.1 Problem Statement

This thesis tackles the usefulness of development data traces in conformance problems that occur in software development teams of open-source projects. Assuring that developers are actually following the defined process is usually a hard task. Similarly, assuring the correct alignment of team members with processes is not easy [5]. However, the adopted software development process should be righteously respected by team members, because of the benefits it delivers, such as high quality end products, cost reduction, better collaboration, etc. Unfortunately, it is common that Product Owners or development teams shift from the prescribed software model to adopting their own, due to various factors, such as their preferences, time-pressure, past experiences, and more. Other reasons which lead to processes getting disrupted along the way comprise inappropriate training, no keen commitment and lack of communication/collaboration among roles [5].

However, in current age of software engineering, agile teams create artifacts and traces in the development environment that are captured by issue-tracking systems. There might be some deviations from the intended software development project, and that these deviations can impact on the results. Considering that process mining has been applied in many areas, including software engineering, we can take advantage of its techniques and capabilities, such as conformance checking, in order to attempt to utilize the data to verify the correct execution of Scrum practices and discover the (non) compliance among the documented Scrum framework.

As mentioned previously, the Scrum framework allows for intra-teams adjustments to certain limits, meaning that different companies might adopt different variations of

Scrum. However, all these companies must at least adhere to the abstract or core model of this framework. Moreover, [7] suggests certain norms that be followed, in order to be a Scrum team. We can map these norms to certain business rules of Scrum, and later use them for rule-based compliance checking.

To that point, this thesis aims to give answers to the following research questions:

- **RQ1: To what extent are data extracted from Jira, useful to apply conformance checking techniques?**
- **RQ2: What are the most common Scrum practices among the development teams from the open-source projects?**
- **RQ3: What are the compliance levels of the development teams towards the implemented Scrum practices?**

1.2 Thesis Outline

The remainder of this thesis proceeds as follows: **Section 2** provides an overview of the current state of the Scrum framework and Process Mining capabilities, with emphasis on Conformance Checking. **Section 3** reveals the state-of-the-art of the related research. In **Section 4** I describe the methodology that is followed along the thesis to reach the thesis goals. In **Section 5**, I report the results obtained from the research, which are further elaborated in **Section 6**. Whereas, in **Section 7**, I summarize the conclusions of the thesis and map out the work to be be conducted in future research endeavors.

2 Background

2.1 Agile Software Development

The rapidly growing field of Software Engineering does not restrain when it comes to introducing new methodologies, as per the last 25 years. The latest methodologies are classified as agile, and while there is not any clear statement to what the word agile refers to in this context, it has undoubtedly gained a lot of attention not only among practitioners, but also in the academic realm [9]. Agile methods are commonly based on incremental development, in which solutions derive from collaboration between teams that are individually organized, but with a common ultimate goal. Methods like Scrum, Kanban, XP are based on the 12 principles, unified in the Agile Manifesto [10]. Initially, the study considered all of the aforementioned methods, however Scrum was finally chosen, due to its higher popularity and trustworthiness among software teams over the other methods. In addition, Scrum carries many benefits compared to other agile methods, such as easier scalability, more flexible to changes, predictable timeframes, higher software quality and more. Finally, while exploring the data, several evidences within the data indicate Scrum usage, as elaborated in more detail in the Methodology section of this thesis.
In the subsections below, the state of the art of the Scrum framework is thoroughly presented. In addition, a brief overview of Process Mining is given, alongside a summary of Conformance Checking types and techniques.

2.1.1 Scrum Overview

According to HELENA[1] and Version One[2], Scrum is the most popular agile framework worldwide. This highly motivated the decision of choosing this framework for this study. Scrum was developed with the purpose of managing system development process more easily. Thus, it starts with the premise that, it is too complex and unpredictable to plan all of the software development in advance. Its main focus is on how software teams ought to function properly, so to ensure flexibility of the system in a dynamic

[1]HELENA is an international study on the use of Hybrid dEveLopmENt Approaches in software systems development. Website: https://helenastudy.wordpress.com/about/
[2]https://stateofagile.com/

environment [9]. As per the originators of Scrum [8], it is rather a lightweight model which aims to facilitate value creation within teams and organizations by employing adaptive solutions for problems of high complexity. Moreover, the Scrum framework is formally documented by its creators in the Scrum Guide. For a successful Scrum implementation, Scrum team members have to embrace certain values, follow established events and interact accordingly with the created artifacts.
To achieve it, an iterative and incremental process should be cultivated. The skeleton of this framework is shown below in Figure 1. The circle beneath encapsulates an iterative instance of the development tasks that take place successively, each of them leading to a shippable increment of the final product. On the other hand, the upper circle shows the daily inspections, during which, developers inspect each others' progress and make relevant adaptations.

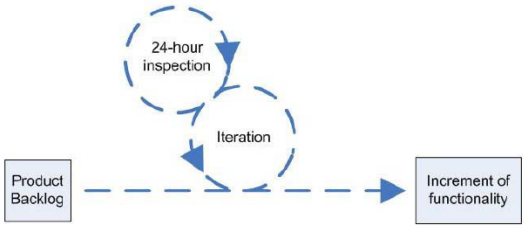

Figure 1. Scrum Skeleton - Figure is taken from [11]

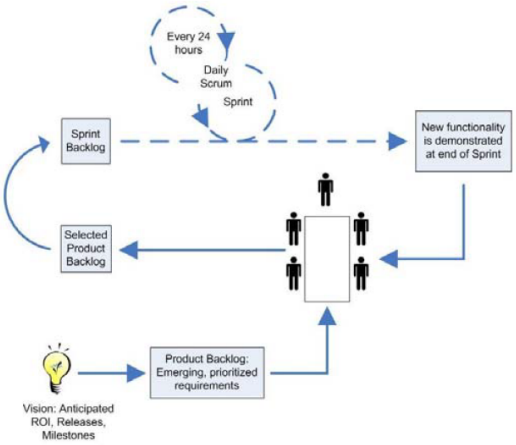

Figure 2. Detailed Scrum Flow - Figure is taken from [11]

For creating valuable increments during each Sprint, the entire Scrum Team is employed. This team represents an underlying unit of Scrum, and it is consisted from a small team of people [7]. Scrum defines three accountabilities within its team, namely:

- **the Scrum Master (SM)**: who is primarily responsible for establishing Scrum as per the Scrum Guide. Therefore, he/she manages the Scrum process by teaching and helping everyone understand Scrum in the organization, ensuring that Scrum is implemented in a way that matches the organization's culture and making sure

that everyone complies to Scrum practices and rules while delivering the expected benefits.

- **the Product Owner (PO)**: who is accountable for generating and maximizing the value of the product, as an end result of the Scrum Team's work. Moreover, the PO is liable for an effective management of the Product Backlog, i.e., a prioritized list of items (requirements in the form user stories) with estimated times in order to transform them into working product functionality.
- **Developers**: are the individuals within the Scrum team committed to delivering usable increments by the end of each Sprint. Developers work as self-organizing, self-managing and cross-functional teams and are collectively in charge for the success of each Sprint and the whole project.

A holistic view of the detailed Scrum flow is presented in Figure 2. Any project developed under the Scrum framework, begins with a vision of the product. By using this vision and extracting all known requirements from stakeholders, the Product Backlog is initially created. It is further prioritized and separated into specific releases, as agreed collectively. All of the proceeding activities are clustered together and labeled as Scrum events, while the Sprint serving as a container for all of these events. The main purpose of conducting the Scrum events is to formally inspect and adapt the Scrum artifacts.
To have a better understanding of Figure 2 and the aforementioned notions, a short explanation for each of them is provided below.

- Scrum Events:

  the Sprint: [7] K. Schwaber and J. Sutherland describe *Sprints* as the heartbeat of Scrum, where tangible value is created from the ideas. For consistency purposes, they are time-boxed to one month or less (depending on team size), and the succeeding Sprint begins right after the conclusion of the previous one. A Sprint carries all the work needed to accomplish the Product Goal, and it can only be cancelled by the Product Owner, in case the Sprint Goal becomes obsolete.

  Sprint Planning: is the initial activity within a Sprint, in which the entire Scrum Team joins forces towards preparation of a plan, laying out the work to be

completed during the specific iteration. This plan targets three topics, respectively: *"Why is this Sprint valuable?", "What can be Done this Sprint?", "How will the chosen work get done?"*.

Daily Scrum: is a 15-minute event, which gathers the developers (in some cases also the SM and PO) at the same time and place on a daily basis. It aims to improve collaboration, identify and alleviate impediments and promote swift decision-making.

Sprint Review: is the second to last event within the Sprint, lasting a maximum of four hours or less. The purpose here is to check the outcome of the Sprint and agree on future adjustments and work. The key stakeholders are welcome to participate and discuss the up to present progress.

Sprint Retrospective: is timeboxed to three hours at most, or less (depending on the whole duration of the Sprint). It finalizes the Sprint lifecycle by encouraging the Scrum Team to revise its development process to increase its effectiveness and quality for the upcoming Sprints.

- Scrum Artifacts: refer to work or created value. Scrum has numerous artifacts, such as:

  Product Backlog (PB): is a list containing all the requirements expressed as User Stories[3] This dynamic artifact is created and managed by the PO, who regularly updates it, according to the changing-requirements.

  Sprint Backlog (SB): consists of the Sprint Goal, a list of specific User Stories or PB items to be treated during the Sprint, and a plan on value/increment delivery. Therefore, SP has all the necessary information to answer the: *Why, What, How* questions, respectively.

  Increment: is referred to as a stepping stone toward the ideal Product Goal. Each finalized increment is usable and adds value to the prior one, while ensuring that they all function properly as a whole. In order for a PB item to become an increment, it needs to meet the Definition of Done (DoD). The latter one, formally

[3]User Stories are concise statements created based on user requirements. They are written in the form of: As a <type of user> I want to <some goal> so that <some action> [12].

describes the state of an increment, and provides transparency by enabling everyone to understand what work was completed as part of the Increment.

To summarize, the Scrum framework is lightweight and simple, however, purposely left incomplete and can sometimes be difficult to master [7].

2.2 Process Mining and its Capabilities

Process mining is a young researh discipline with the objective of discovering and improving real processes. It achieves its targets by extracting the multitudes of data from available event logs in today's ubiquitous systems. These data are then utilized to furnish us with better insights on the actual processes, detect and analyze deviations and improve the quality of the process models [13]. There are three capabilities of Process Mining: Process Discovery, Conformance Checking and Process Enhancement. Figure 3 shows the positioning of these capabilities against the data and the real world.

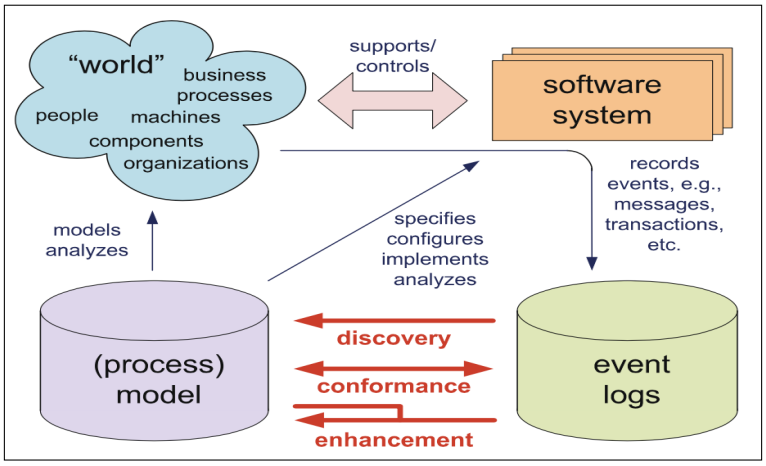

Figure 3. Process Mining Capabilities - Figure is taken from [13]

2.2.1 Conformance Checking

Conformance Checking or Compliance Analysis is one of the capabilities of Process Mining. In order to carry out conformance checking, an existing process model is compared against an event log of the same process model. Conformance Checking techniques take as an input an event log and either a Process Model and more often, a set of business rules that should be fulfilled by the event log. Therefore, there are three types of Conformance Checking [14]:

1. **Rule-Based Conformance Checking:** *Checking that a process follows certain rules.*
2. **Model-Based Conformance Checking:** *Comparing the behaviour captured in the process model against the one in the event log.*
3. **Exceptional Behavior Analysis:** *Detecting anomalous behavior.*

Additionally, there are three types of Conformance Checking techniques [13] as summarized in Table 1.

Table 1. Summary of Conformance Checking Techniques

Conformance Checking Technique	**Description**
Conformance Checking using Causal Footprints	Causal footprints are very flexible, in the sense that we can compare logs to logs, models to models and logs to models, by creating and comparing the footprint matrices. It also implicitly captures fitness, precision and generalization but in an indirect way. The approach has some limitations, such as: 1. Frequencies are not used. 2. Behavior is only considered indirectly. 3. Aims to capture fitness, precision, and generalization in a single metric.
Conformance Checking based on Token-Based Replay	This approach is based on replaying the event log to diagnose differences between the model and the observed behavior. It is a more refined method compared to the first one. Through replay we can capture diagnostic information - where are the differences and how severe they are and we can also compute a global fitness metric. Limitations of the replay approach are as follow: 1. Basic replay approach assumes visible and uniquely labeled transitions. 2. Local decision making may lead to misleading results.

Table 1 continued from previous page

Alignment-Based Conformance Checking	This is the most advanced approach relating the event logs to process models. Advantages of aligning the log and model: 1. Observed behavior is directly related to the modeled behavior. 2. Very flexible (any cost function, assigning penalties). 3. Provides detailed diagnostics.

Taken altogether, these techniques are used to check if reality, as captured in the log, conforms to the rules or model, and vice versa [13][14].

2.2.2 Process Model of Scrum

In this section, the process model of the Scrum framework is shown. It was modeled using the Camunda platform[4], which among many purposes, serves as a BPMN process modeler. The abstract model of the framework was initially modeled, containing only the main tasks (events, artifacts, etc.). This initial model depicted in Figure 4[5] describes the general workflow of Scrum, i.e. its main tasks. A brief textual description for these tasks/steps is provided below:

1. **Product Vision**: refers to coining the product vision and composing the Scrum Team.
2. **Product Backlog Creation**: preparing the backlog items to be completed and delivered during their respective sprint.
3. **Product Backlog Prioritization**: assigning priorities to the identified product backlog items.
4. **Sprint Planning**: collaborating and discussing the high-priority work for the upcoming sprint. Also includes defining the sprint goal.
5. **Sprint Execution**: refers to the work done during the sprint, towards meeting the sprint goal.
6. **Daily Scrum**: inspecting the progress toward the sprint goal, done on daily basis.

[4]https://camunda.com/products/camunda-platform/modeler/
[5]Figure 4 and Figure 5 are adapted from [2].

7. **Sprint Review**: discussing the work completed during the sprint.
8. **Sprint Retrospective**: checking what went right and identifying areas for improvements in the next sprints.

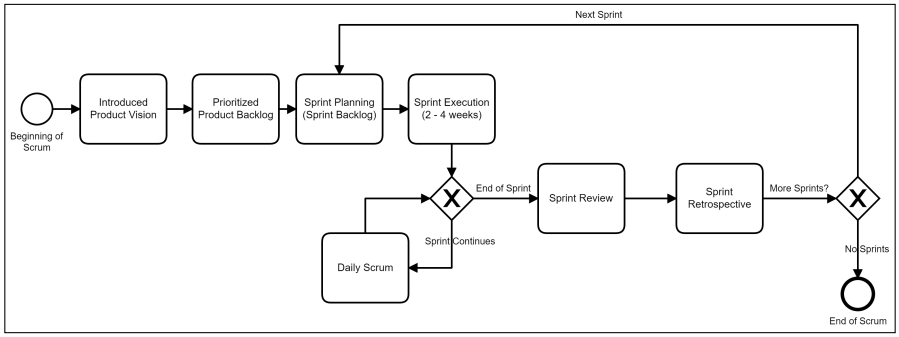

Figure 4. Abstract Process Model of Scrum

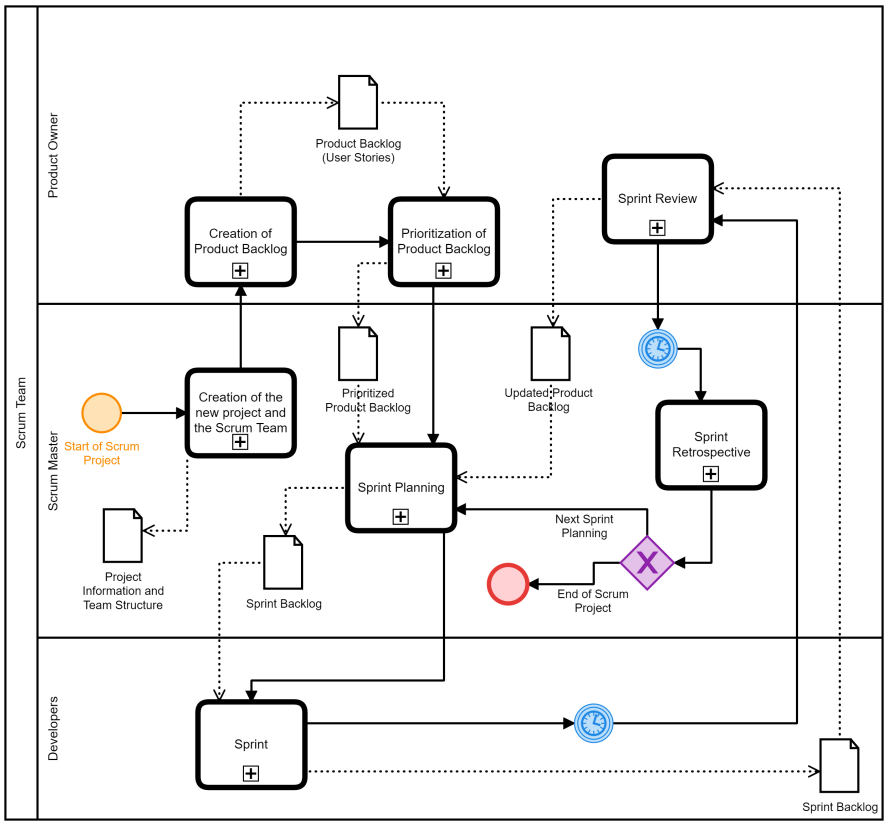

Figure 5. Detailed Process Model of Scrum

Irrespective of the fact that each of the tasks in Figure 4 is of an abstract type, they are all composed of a sequence of other tasks. The detailed model in Figure 5 extends and improves the abstract model by defining the participants through pools and lanes. Additionally, the abstract tasks have been transformed to call activities, invoking their predefined subprocesses. The updated model is shown in Figure 5. Here, the *Scrum Team* pool is defined, which contains three lanes. Each of these lanes corresponds to the three accountabilities proposed by the framework, namely: *Product Owner, Scrum Master and Developers*, and encompasses the tasks that should be performed by each respective role. When describing Scrum in the previous section, the Scrum Artifacts were also mentioned, which store the necessary information that is updated iteratively during the various Scrum events. To represent these artifacts, data objects are added to the abstract model, which allow for data sharing between the tasks, necessary for execution of the next tasks and more efficient team communication overall.

3 Related Work

Although Business Process Mining is a nascent field, it has already found applicability in several domains. According to [15], Process Mining stands at the intersection of data mining and computational intelligence in one side, and process analysis and modelling in the other side. There have been several attempts on using Process Mining Techniques for automatic process discovery [16]. Besides, Process Mining has been used in the educational realm [17] and [18]. Furthermore, the conformance checking capability of Process Mining has been successfully implemented in the healthcare domain [6], [19] and [20].

On the other hand, the idea of combining BPM and PM with Agile Software Development Methods, also exists, as can be seen below.

One of the first attempts to integrate Process Mining techniques into the Agile Development Lifecycle is done in [21]. This work proves that Process Mining can be successfully embedded within agile models and can significantly benefit the software design and development.

The research conducted in [22] states that although Process Mining Techniques have been used in business process domains, they have not found a lot of implementation on the software engineering processes yet. This work employs Process Mining Techniques

to discover the implemented process model by a Scrum Team. It successfully shows that such techniques are capable of discovering and verifying process deviations as well as the divergence within the work items workflows.
The compliance analysis of three ASDM-s against the ISO/IEC 29110 standard by Galvan, et al. [23] shows that some Agile methods present higher compliance levels with the aforementioned standard than others. Concretely, they studied the compliance problem of the cases of Scrum, XP and UPEDU versus the Project Management process of ISO/IEC 29110. Both the models and the standard point to small teams or companies. Their study analyzed the categories of roles, activities and artifacts, by assigning values from 1 (low) to 3 (high) to assess the compliance levels of the ASDM-s against the ISO/IEC 29110 standard. The findings reveal that even though Scrum and XP are more popular Agile methods, UPEDU and SCRUM showed high compliance with the standard, whereas XP reported a moderate compliance level.
The study of El Kharbili et al. [24] gives a comprehensible outlook on the state-of-the-art of compliance checking techniques, with a focus on the BPM field. The authors reviewed important work done in this direction and they conclude their literature review by classifying the existing techniques into two main categories:

- **Forward-compliance checking**: an approach with a preemptive nature. Such techniques check the compliance of Business Processes before and during run-time/execution. Thus, they are suitable for detecting non-compliant behavior and prevent it from further happening. These techniques can be categorized again, as:

  1. Design-time compliance checking
  2. Run-time compliance checking

- **Backward-compliance checking**: also known as post-execution compliance checking. Though unable to prevent non-compliant behavior from occurring, these techniques use graphical notations for modeling the business processes, making them very useful for people who use them (e.g. business analysts).

The authors conclude that there is a need for more empirical research to be done as to create compliance techniques that cover the full BPM lifecycle.
In [2], Zaouali and Ghannouchi propose a prototype aiming to help teams conduct ASD through Scrum, based on BPM. They have chosen Scrum, as it was the most popular

framework at the time of their research. After modelling the Scrum framework on a BPMS platform, they implemented an executable prototype for teams that operate with Scrum. The authors claim that their contribution aims to automate several facets of Scrum, but not the whole Scrum process. This work concludes that, using an (partially) automated information system for managing Agile methods (Scrum), delivers numerous advantages, such as: fostering team communication and collaboration, guidance, training, continuous improvement, etc.

Moyon et al. [25] have undertaken a slightly different approach to ASD Compliance problem, however, insightful for this thesis. Their work focuses on the security compliance of Agile methods, but they emphasis that apart from security, compliance to predefined workflow, is a critical issue too, when it comes to applying agile methods in regulated environments. This paper proposes a method which will help organizations accomplish continuous secure development according to the IEC 62443-4-1 standard requirements. They state that by mapping these requirements into an agile model, iterative secure development can be attained. They acknowledge the popularity of Scrum, but since they target large organizations, their agile model of study case is Scaled Agile Framework (SAFe). This approach is based on the creation of mergeable models of the aforementioned process standard and agile method through BPMN. There are three main steps they follow, respectively: (1) Creating spearate visual models (for both process and standard) in BPMN, (2) Validating the models by collaborating with experts and field practitioners, (3) Merging the models, or with other words: extending the process model with the standard model.

Other related work is the contribution by Gallina, Muram and Ardila [26]. Their work accentuates the need for clarifying or rethinking the existing ways/techniques for compliance checking. The authors target hybrid software development methods (manifold Scrum versions), due to the fact that pure Scrum is not suitable nor sufficient for safety-development environments, as is the scope of their research. Anyhow, they discuss about augmented variations of Scrum, such as R-Scrum and Safe Scrum and the four means of managing process compliance. These include: (1) Compliance through mapping tables, (2) Argumentation about compliance, (3) Automatic compliance checking, (4) Ontology-based mapping. Lastly, they conclude their work by analyzing the various use cases stemming from differing domains, understanding the used compliance means that are being implemented, in order to create a vision on how to apply such compliance

means in the context of hybrid development methods.
Analyzing developers' agile behavior from the data and event logs they produce from using their IDE is done in [27]. This research uses process mining techniques and machine learning, including conformance checking practices in order to infer insightful results. Their findings support and further consolidate the discovery of coding patterns and developers' (non)compliant behaviors to (from) the prescribed software processes.
Another useful work that takes advantage of the versatility of Process Mining techniques for compliance checking of software processes is done by Lemos et al. [5]. The authors of the paper have applied process mining to conformance checking on a database with more than 2000 event logs generated (and provided) by a Brazilian software company. The software development process in the company is defined internally. The study focuses on the control-flow perspectice of process mining. Their endeavors conclude that process mining techniques can effectively serve for management and improvement of software development processes within software organizations.
Matthies et al. [28] present ScrumLint in their paper. This tool analyzes development artifacts (commits, tests), and based on the defined conformance metrics, it is able to detect variations and violations of the executed development process artifacts, thus giving fast feedback to developers and enabling areas of improvements in team workflow.
The reviewed work reveals that similar approaches and objectives to the ones of this thesis have been previously carried out. Process mining has been applied for Compliance Checking on software process models other than Scrum, using data extracted from IDE-s. On the other hand, Scrum has been linked to BPM in some studies, in order to automate parts of it. A summary of the related work can be found in Table 2.

Table 2. Related Work

Study Name	**Source**	**Study Area**	**Used Method**
A Compliance Analysis of Agile Methodologies with the ISO/IEC 29110 Project Management Process	[23]	Evaluating compliance levels of 3 ASDM with a software development standard	IDEF0 conceptual tool

Table 2 continued from previous page

Towards Continuous Security Compliance in Agile Software Development at Scale	[25]	Continuous Security and Compliance Problem in ASD	Model-based approach review
Compliance of Agilized (Software) Development Processes with Safety Standards: a Vision	[26]	Process Compliance in hybrid ASDM-s	Literature review, empirical-based vision
Software Development Process Mining: Discovery, Conformance Checking and Enhancement	[27]	Cloud-based IDEs, Mining software processes	Data Extraction and Analysis, Process Mining, Machine Learning
Business Process Compliance Checking: Current State and Future Challenges	[24]	Compliance Checking Techniques in BPM	Systematic literature review
Proposition of an approach based on BPM to manage agile development processes	[2]	Automating Scrum processes based on BPM	BPM, Systematic literature review
Using Process in Software Development Process Management: A Case Study	[5]	Process Mining, Conformance checking	Control-Flow based Process Mining, Data analysis

Table 2 continued from previous page

ScrumLint: Identifying Violations of Agile Practices Using Development Artifacts	[28]	Agile practices, Violations of practices, Conformance	ScrumLint Tool
Agile Development with Software Process Mining	[21]	Process Mining, Agile Methods, Software Process	Mining User Behavior, Mining Runtime Traces
The Discovery of the Implemented Software Engineering Process Using Process Mining Techniques	[22]	Process Mining, Mining Techniques, Software Process	Process Discovery, Mining Techniques ProM

Taking into account the similar endeavors done previously, it is not of our knowledge that process mining techniques nor capabilities are applied to verify the software teams' adherence to the real Scrum framework, and not to a variation of it. Moreover, using data obtained from issue-tracking software (Jira), mapped out to and verified against a collection of original Scrum rules extracted from relevant Scrum sources, mainly the Scrum Guide, is a novel approach that furthermore distinguishes this study from other related work. Hence, this remains the main contribution of this thesis, which I believe supplies useful findings targeting the input data that is needed to acquire higher compliance to the Scrum framework, that can furthermore be mapped into valuable insights for development teams to improve their overall agility or Scrum adherence.

4 Methodology

4.1 Study Design

In this section, the study design and the dataset used for this research are described extensively. Figure 6 depicts an overview of the study design. It begins with data acquisition from ten publicly available Jira issue-tracking software instances. As mentioned previously, these data come from real-life open-source projects and the dataset was provided by the supervisors of the thesis. Simultaneously, the Scrum Guide has been carefully examined and studied in order to define a set of rules, where each rule relates to or denotes a Scrum practice at different levels of granularity. A detailed analysis has been done to define the data fields that are needed to verify each rule programmatically. Thus far, the first research question is addressed. Proceeding forward, I implemented the computable or verifiable rules using the data of each project in order to answer the remaining research questions, which target the popularity of the practices and the conformance of the projects against these practices/rules, respectively.

Notably, the data fields and the information contained within the issue reports seem to indicate that these projects were incorporating agile practices during the development process. For instance, the issues of type *'story'* follow the standard syntactical form. The following issue is taken from the Jira logs of the XD project: *"As a developer, I'd want to document the limitations of HSQL DB when using composed jobs."*. This, along other indicators observed in the data, such as the existence of sprints, well described issues, etc. serve as a further motivation for this study.

The datasets used in this study, besides the coding scripts for cleaning the data and verifying the Scrum rules programmatically, can be found in this publicly accessible GitHub repository [6].

[6] *https://github.com/mirlindm/MasterThesis-Mirlind-2021*

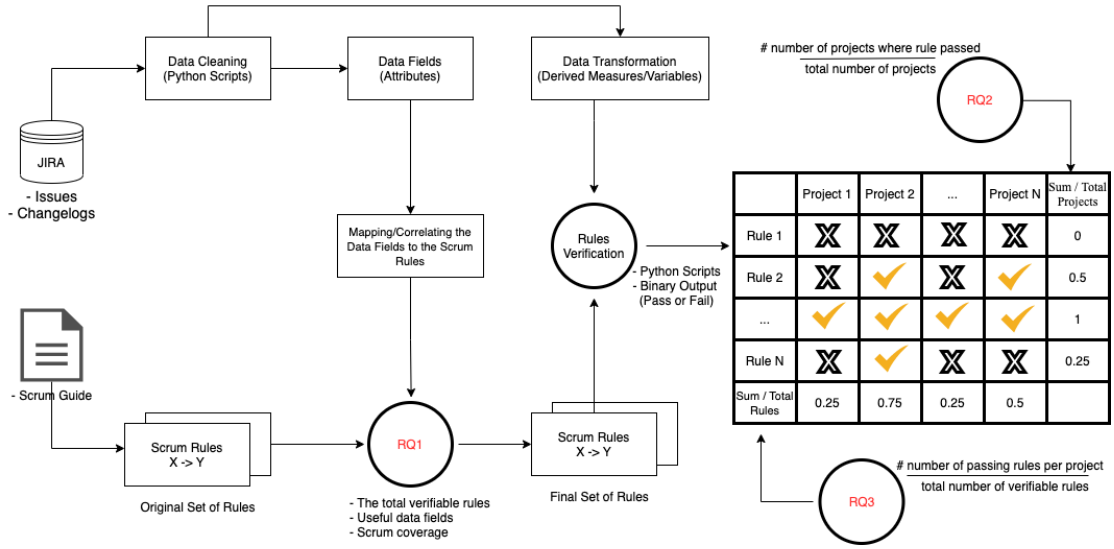

Figure 6. Study Overview

4.2 Research Objective and Research Questions

The main research objective of this study is to identify what data or input taken from Jira, are useful to apply conformance checking in the context of Scrum software development processes. Complementary, other objectives focus on uncovering the Scrum practices that are more popular among the software teams, and on the other hand, bring to light insights on how compliant are they towards the framework itself. The derived understandings from these objectives are crucial and give directions for future research in the realm.

1. *RQ1: To what extent are data extracted from Jira, useful to apply conformance checking techniques?*
   Motivation:

   Software development teams use issue-tracking systems to capture and store these data in a single place. Within the data recorded in trackers like Jira, a lot of information about the development processes that teams follow, is contained too. For instance, the dataset used for this study contains data fields carrying information about the Jira issues, such as the type, description, the developer it has been assigned to, the sprint it belongs to, as well as information about the changelog of the issue status over its workflow, etc. This indicates that there is a plethora

of information that can be collected from these trackers. However, we need to keep in mind that open-source projects share different characteristics compared to commercial products, since they are built on open-collaboration, where the number of software practitioners involved changes along the way, leading to a variable team composition and data that can be tricky to infer insights from. Considering this, one important question that arises is, what kind of data is useful to apply conformance checking in the domain of agile software development among open-source projects? Moreover, narrowing it down to the Scrum framework as the agile framework chosen for the study, which of the Scrum practices are we capable of verifying through the data, and what is the overall coverage of the framework that we can check using the data. This research question aims to give answer to all of the aforementioned inquiries.

Procedure: The main challenge for this research question is mapping the obtained data to the real Scrum practices. To answer this question, I began by manually creating a set of Scrum rules, based on the Scrum Guide [7]. I relied on the Scrum Guide mostly, because of its softened prescriptive language, which makes it easy to read and understand, it is created by the originators of the framework itself. I assume the projects are following Scrum, since there are numerous indicators in the data that support these assumptions. Using the Scrum Guide, I extracted a set of rules in which, each rule corresponds to or reflects a Scrum practice as defined officially. Hence, the notions of rules and practices in the Scrum context are used interchangeably in this thesis. Each of the rules are then mapped to a specific sentence or section in the Scrum Guide. I labelled each rule according the phase it belongs (pre-game, game and post-game). Moreover, additional information are added about the roles that are involved in each of the rules (SM, PO and developers), along with the information about the Scrum Event or Artifact that is affected by the rule (Daily Scrum, Product Backlog, etc.). With the set of rules in place, I have added a pseudo-algorithm for each rule, which helped to identified the derived variables, that are checked for availability within the data fields from Jira. This way, I calculate:

(a) *The total number of rules that are possible to check with the data.*

(b) *Which data fields are useful to check for the rules*

(c) *The Scrum Events and Roles for which the data is not enough to check the respective rules*

(d) *How much of the Scrum framework are we able to inspect or check through the rules, i.e. Scrum coverage*

These calculations cover and answer all of the question marks put forward by the research question.

2. *RQ2: What are the most common Scrum practices among the development teams from the open-source projects?*
   Motivation: Assuring that developers are actually following the defined process is usually a hard task. However,the adopted software development process should be righteously respected by team members, because of the benefits it delivers, such as high quality end products, cost reduction, better collaboration, etc. Unfortunately, it is common that product owners or development teams shift from the prescribed software model to adopting their own, due to various factors, such as their preferences, time-pressure, past experiences, and more. Moreover, we are not sure about the specific practices that the teams were following. Since developers are constantly devising improved ways of developing quality software [5] and on the other hand, the Scrum framework allows for adaptive modifications to it [7], through this research questions, the aim is to obtain more information about the most common Scrum practices that are present within the development activities of the teams. On the other hand, we believe that some of these practices are more popular than others, for instance: beginning and ending the sprints and tasks on time, planning the sprints and the sprint backlog items. This is known through previously conducted surveys, such as Helena [29], and Version One[7]. Anyhow, we do not know how consistent these findings are with our data. Thus, this research question assists on ranking the practices according to their popularity in the projects.
   Procedure: Given a definition of Scrum, a set of rules and data collected from Jira, we can employ some steps to check whether the process reflected by the data, conforms to the Scrum rules/practices. The steps undertaken to answer this

[7]https://stateofagile.com

research question are dependent on RQ1. Taking into account the set of Scrum rules defined in the first research question, we can check whether each of the rules passes or fails on the given project data. Then, for each rule, its final result (i.e. how common the rule was among the projects) is calculated through the following formula:

$$RQ2forEachRule = \frac{numberOfProjectsWhereRulePasses}{totalNumberOfProjects}$$

Lastly, after calculating the results (range from 0 to 1) and reporting them as percentages for each rule from the set, we create a ranking of the practices based on the overall percentages, where I begin by reporting the practices with higher percentages, i.e. more popularity.

3. *RQ3: What are the compliance levels of the development teams against the Scrum practices?*
   Motivation: In RQ1, we focus on the data and investigate how useful they are for conformance checking towards Scrum. In RQ2, we discover the most commonly used Scrum (agile) practices within the software teams. In the utopian situation, all of the development teams comply flawlessly with the normative process established in the company. However, teams often deviate from their processes and, therefore, some teams show higher compliance than others. Similarly, with the project data we have, we do not know to what extent do the software teams righteously execute Scrum's guidelines. That being so, through this research question, we rank the software teams of our case study as regards their compliance levels to the Scrum framework.

   Procedure: We achieve the insights from this research query through a comparative analysis between the projects. Again, given a definition of scrum, a set of rules, and data collected from Jira, we can check if the process that the data reflects conforms to the specification of Scrum. The steps undertaken to answer this research question are dependent on RQ1. Taking into account the set of Scrum rules defined in the first research question, we can check whether each of the rules passes or fails on the given project data. Then, for each project, its compliance is calculated through the following formula:

$$RQ3 for Each Project = \frac{number Of Passing Rules Per Project}{total Number Of Verifiable Rules}$$

The calculated value represents the degree of compliance, where 0 means no compliance at all, whereas 1 shows a full compliance of the given project. These values are reported as percentages (range from 0% to 100%) in the Result section. Using these results, we conclude this research question by creating a ranking for each project based on the individual percentages, where I begin by reporting the projects showing higher compliance to the framework, to the lowest.

4.3 Datasets

The datasets that are used for the purpose of this study capture the development activities in the form of issues of 10 open-source projects. There are previous studies incorporating the same data of these projects [30] [31]. A more detailed description of the projects is shown in Table 3.

Table 3. Open-Source Projects Description

Project Name	**Description**	**Website, Jira and GitHub URL**
XD - Spring XD	A unified and distributed service tackling big data complexity	projects.spring.io/spring-xd/ jira.spring.io/ github.com/spring-projects/spring-xd
APSTUD - Aptana studio	Web Application IDE	www.aptana.com/ jira.appcelerator.org/ github.com/aptana/studio3
TISTUD - Appcelerator Studio	Platform for native apps, mobile APIs, real-time analytics	appcelerator.com/ jira.appcelerator.org/ github.com/appcelerator
MOBILE - Moodle mobile	Mobile Application for Moodle	moodle.com/app/ tracker.moodle.org github.com/moodlehq/moodleapp
MDL - Moodle	An open-source learning management system	moodle.com/lms/ tracker.moodle.org github.com/andrewnicols/moodle

Table 3 continued from previous page

DNN Platform	.NET CMS Software	www.dnnsoftware.com/ dnntracker.atlassian.net/ github.com/dnnsoftware/Dnn.Platform
MESOS - Cluster management software	A distributed systems kernel	mesos.apache.org/ issues.apache.org/ github.com/apache/mesos
MULE - MuleSoft integration platform	Integration platform for SOA, SaaS, and APIs	www.mulesoft.com/ www.mulesoft.org/ github.com/mulesoft/mule
NEXUS - Nexus artifact manager	Repository Manager	www.sonatype.com/nexus/repository-pro issues.sonatype.org/ github.com/sonatype/nexus-public
TIMOB - Titanium Command Line (CLI)	Open source application development platform	jira.appcelerator.org/projects/TIMOB/ issues/TIMOB-28360?filter=allopenissues jira.appcelerator.org/ github.com/appcelerator/titanium_mobile

The datasets originally contained 17 data fields, however not all of them are useful to be employed for checking the Scrum rules. Therefore, 7 data fields were removed from the dataset completely, thus retaining only the data fields that carry valuable information for the purpose of the research. The fields along a short description are summarized in Table 4.

Table 4. Data Fields Description

Data Fields	**Description**
Data fields that describe the Scrum teams, projects and sprints	
assignee.name	This field indicates the developers working within the projects.

Table 4 continued from previous page

creator.name	This field indicates the user who has created the issue. By default, this field should contain a value.
reporter.name	This field indicates the reporter of the issue. Issue reporting can be done by developers themselves or by the Product Owner, therefore we can't use this data field to check for the accountabilities of either role.
project	This data field indicates the name/ID of the open-source project.
sprint	This data field indicates the unique identifier of the sprints.
key	This field indicates the unique identifier of the issue.
issuetype.name	The type of issue report (e.g Bug, Story, Improvement, Task).
summary	This field provides textual description of the issue. More specifically, it indicates the title of the issue report.
description	This field provides textual description of the issue. It can also be viewed as the body of the issue.
priority.name	This field indicates the priority assigned to the issue.
storypoints	This field indicates the amount of story points for each issue.
Data fields that describe the sprints and issues timewise	
created	This field indicates the date when the issue was created.
resolutiondate	This field indicates the date when the issue was resolved.
Data fields that describe the statuses associated with the issues	
status.name	This field indicates the status of the issue (e.g. TODO, DOING, In Progress, etc.).
status.id	This field indicates the ID assigned to each status name.
status. statusCategory. name	This field provides extensive description of the status.name field.

The data fields that were removed from the dataset, due to are: *components, fixVersions, issuetype.subtask, versions, watches.watchCount, resolution.description, resolution.name*

4.4 Data Cleaning Steps

Although the Scrum framework allows for modest and limited modifications, it still defines numerous strict rules that the entire Scrum team presumably adheres to. In order to map these rules to the development activities of the teams, captured in a single dataset, we first need to make sure that the data is clean, qualitative and allows for further calculations in the scope of our study.
The data cleaning was done in iterative steps. Individual projects have been exported to separate datasets, in order to be analyzed in isolation. This is to ensure whether we can check for the defined Scrum rules in all projects separately, considering that the data differs among the projects, in the same way as the development process differs among the teams. Moreover, certain data fields in certain projects need to be analyzed differently. In any case, there are some cleaning and pre-processing steps that were applied to all projects:

1. **Column-related cleaning:** The following columns were removed from the datasets, since they did not contain any valuable information for this study.
    - components
    - fixVersions
    - issuetype.subtask
    - versions
    - watches.watchCount
    - resolution.description
    - resolution.name
2. **Issue-related cleaning:** Issues that met the following criteria were removed:
    - Duplicate Issues
    - Issues having no creator associated to them
    - Pointless issues, i.e. issues having no issue title or body
3. **Project-based cleaning:**

As mentioned previously, the open-source projects' data are analyzed separately, rather than retaining them into one large dataset, as was initially provided. This way, we can obtain clearer insights on the potential discrepancies that might be present within the data. Another reason for separating the open-source data, is that different projects have followed different Scrum practices, and the data that reflects it, can vary significantly from one project to another.

These missing data convey useful information to us.

- Projects such as **MDL, Timob, Mobile, XD and Tistud** have many missing values persisting in the sprint data field.
- We can also see that there are many issues among the projects that were never assigned any developers (*assignee.name*).
- There are many issues that were never completed (*resolutiondate*)
- MDL and Mobile projects have almost no values in the *Storypoints* field.

However, these missing values are kept, as valuable information would be lost otherwise, which supports our rule-based conformance checking of the projects.

5 Results

In this section, the previously defined research questions are answered. It begins by providing insights on the data, before and after the data cleaning steps are applied. Thereafter, I describe the rules and then I present the results, which are organized by the research queries and are discussed more thoroughly in the Discussion section. The underlying motivation of this analysis is to check whether and to what level we can map the data produced by agile development activities to the rules reflecting Scrum practices. In total, there are 906 sprints that have been run and executed by a total of 824 developers, in order to implement 83223 issue reports. Since team work differs from one project to the other, we can notice that there are several issue statuses, varying according to the projects. A more detailed look of the projects information, such as issues, sprints can be found in Table 5.

5.1 Research Question 1

In order to answer *RQ1: "To what extent are data extracted from Jira useful to apply conformance checking techniques?"*, an original set of 39 rules have been identified[8], out of which, 26 rules have been manually extracted from the Scrum Guide [7], where each rule corresponds to a prescribed Scrum practice. Moreover, there are 13 additional proposed rules that are not directly reflected in the Scrum Guide documentation, however, they are derived from the combination of other rules or inferred from empirical usage of Jira and Scrum. For instance, the Scrum Guide states that the Scrum team should remain small and nimble, up to 10 members. On the other hand, it also states that there should be only one Product Owner and Scrum Master each. Therefore, the development team itself should be composed of around 8 developers. However, this is not explicitly defined in the Scrum Guide, rather, it can be inferred through the amalgamation of multiple other statements.

The tables below present a step by step overview of the undertaken process. Table 7 shows the interpretation or description of the rules. Table 8 presents the original source in the Scrum Guide where the rule was taken from, whereas Table 9 shows the data fields from our dataset, which are used to verify the rules. Going further, from Table 10 we can

[8]The detailed set of the Scrum Rules can be accessed via the following URL: *https://github.com/mirlindm/MasterThesis-Mirlind-2021/tree/main/RQ1%20Results*

view information on the rules that are verifiable and non-verifiable through the data. The data fields that are rendered as most useful for verifying the rules are shown in Table 11. Finally, Table 12 provides information on the Scrum events, roles and artifacts, for which, the available data is not sufficient for verification.

Table 5. Projects details before data cleaning

Project Name	**Total Sprints**	**Total Issues**	**Total Developers**	**Issue Statuses**
XD	66	3691	31	To Do, In PR, Done, In Progress
APSTUD	34	886	11	Closed, Resolved, Open, In Review, Reopened
TISTUD	60	2870	32	Reopened, Closed, Resolved, Open, In Review, In Progress
MOBILE	23	3273	29	Open, Closed, Waiting for integration review, Development in progress, Waiting for testing, Waiting for peer review, Reopened, Resolved
MDL	269	64795	525	Open, Closed, Reopened, Waiting for peer review, Waiting for integration review, Development in progress, Tested, Waiting for testing, Peer review in progress, Integration review in progress

Table 5 continued from previous page

DNN	103	1894	12	Closed, Open, Reopened, Pull Request Submitted, Resolved, Planned Development
MESOS	70	1472	74	Open, Reviewable, Accepted In Progress, Resolved,
MULE	104	1281	39	Closed, Resolved, To Do, In Progress, Reopened
NEXUS	70	1071	22	Open, Refine, Closed, Done, Raw
TIMOB	107	1990	49	In Progress, Open, Resolved, In Review, Closed, Reopened

After performing the data cleaning steps, the final number of issues per project reduced accordingly. In Table 6, a more detailed information is presented. The last column indicates the number of developers that contributed towards the projects. However, not all of the developers have been actively contributing in each sprint of the project. The current information derives from the fact that open source projects employ many collaborators and external contributors who are not part of the team. Later on, during the analysis of the rules, the number of active developers who are part of the development teams of each project has been calculated and shown with the results for RQ2.

Table 6. Project's details after data cleaning

Project Name	**Total Sprints**	**Total Issues**	**Total Developers**
XD	66	3152	31
APSTUD	34	825	11
TISTUD	59	2826	31
MOBILE	23	3242	28

Table 6 continued from previous page

MDL	269	63273	522
DNN	103	1874	12
MESOS	70	1365	74
MULE	104	1278	38
NEXUS	70	1037	22
TIMOB	63	1853	49

Table 7. Scrum Rules Interpretation

Rule ID	**Rule Explanation**
R1	No more than five weeks should elapse for a single sprint.
R2	The number of the Scrum Team members per project should not be largely less or more than 10.
R3	There should be only three different roles employed within a Scrum Team.
R4	The duration of all sprints should follow similar pace.
R5	The next Sprint execution should begin only after the previous Sprint's resolution.
R6	There should be a project clarity identifier attached to each issue.
R7	No considerable amount of time should elapse between the finish of a sprint and the beginning of the new sprint.
R8	There should not be a considerable amount of time for a developer to volunteer and start a new issue after she/he has completed the previous one.
R9	Active members of the development teams should be included in additional activities, other than development.
R10	Daily Standups/Scrums should take no more than around 15 minutes.
R11	Per each Sprint, there should be a constant time and place when/where the daily standup-s take place.
R12	The Sprint Review event should take no more than around 4 hours for longer Sprints (one month), and even less for shorter Sprints.

Table 7 continued from previous page

R13	All Sprint iterations/increments should be over by the time the Sprint Review begins.
R14	All Scrum events should be over by the time Sprint Retrospective begins.
R15	The Sprint Retrospective event should take no more than around 3 hours for longer Sprints (one month), and even less for shorter Sprints.
R16	There should be an agreed Definition of Done for each increment.
R17	Spring Backlog contains the Sprint Goal, PBIs and PBI implementation plan.
R18	There should be at least one Increment deriving from a sprint.
R19	The Product Backlog Items should fulfill the Definition of Done in order to be considered as an Increment.
R20	Scrum Team members and mostly developers should adhere to the organization-wide agreed DoD for the PBIs implementation.
R21	In case of more than one collaborative Scrum Teams, cross-team members should adhere to the mutually agreed DoD.
R22	An Increment may be delivered to stakeholders prior to the end of the Sprint.
R23	No more than one Product Owner should be employed in the Scrum Team.
R24	The backlog does not contain meaningless or empty issues. I consider meaningless issues the issues that belong to no project and have no issue body, description, start and end time and other details that are relevant to developers and other roles. Product Backlog Items should be understandable, therefore should contain a clear description, name, priority, and be identified correctly.
R25	Product Owner and/or Scrum Master should contribute towards the Sprint Goal in order to be part of Daily Scrum meetings.
R26	In case of cancelled Sprints, Product Owner should be the only role within the Scrum Team that can do that.
R27	No more than 8 active developers should be involved in development tasks.
R28	The status of issues should always follow the agreed workflow, depending on the project and development team.

Table 7 continued from previous page

R29	There should not be any unexpected or random person involved at a rather strange part of the Scrum activities, phases or events.
R30	No previously resolved issue should reappear in a future sprint.
R31	There should be a unique identifier/name associated with each Sprint.
R32	There should be a type, such as bug, improvement or task, associated to each issue.
R33	There should be a sprint identifier attached to each issue.
R34	There should be a unique identifier associated with each issue.
R35	There should be timestamp indicating the sprints kick-off.
R36	There should be timestamp indicating the sprints completion.
R37	There should be a minimum of one issue, representing a Sprint Backlog Item, per each Sprint.
R38	There should be timestamp indicating the issue development kick-off.
R39	There should be timestamp indicating the issue development completion.

Table 8. Scrum Rules as per the Scrum Guide and Relevant Literature

Rule ID	**Rule Source in Scrum Guide**
R1	"Sprints are fixed length events of one month or less." (Schwaber and Sutherland, 2020, p. 7, para. 3)
R2	"The Scrum Team is small enough to remain nimble and large enough to complete significant work within a Sprint, typically 10 or fewer people." (Schwaber and Sutherland, 2020, p. 5, para. 3)
R3	"The Scrum Team consists of one Product Owner, one Scrum Master and Developers." (Schwaber and Sutherland, 2020, p. 5, para. 3)
R4	"Working in Sprints at a sustainable pace improves the Scrum Team's focus and consistency." (Schwaber and Sutherland, 2020, p. 5, para. 4)
R5	"The Scrum Master serves the Scrum Team in several ways, including: Ensuring that all Scrum events take place and are positive, productive, and kept within the timebox." (Schwaber and Sutherland, 2020, p. 6, para. 8)

Table 8 continued from previous page

R6	"The Product Owner is also accountable for effective Product Backlog management, which includes: Developing and explicitly communicating the Product Goal; Creating and clearly communicating Product Backlog items; Ordering Product Backlog items; Ensuring that the Product Backlog is transparaent, visible and understood." (Schwaber and Sutherland, 2020, p. 5, para. 1) "Product Backlog refinement is the act of breaking down and further defining Product Backlog items into smaller more precise items. This is an ongoing activity to add details, such as a description, order, and size. Attributes often vary with the domain of work." (Schwaber and Sutherland, 2020, p. 10, para. 9)
R7	"A new Sprint starts immediately after the conclusion of the previous Sprint." (Schwaber and Sutherland, 2020, p. 7, para. 5)
R8	"The Developers are always accountable for: Adapting their plan each day toward the Sprint Goal." (Schwaber and Sutherland, 2020, p. 5, para. 7)
R9	"The Developers are always accountable for: Creating a plan for the Sprint, the Sprint Backlog." (Schwaber and Sutherland, 2020, p. 5, para. 7)
R10	"The Daily Scrum is a 15-minute event for the Developers of the Scrum Team." (Schwaber and Sutherland, 2020, p. 9, para. 4)
R11	"Daily Scrum is held at the same time and place every working day of the Sprint." (Schwaber and Sutherland, 2020, p. 9, para. 4)
R12	"The Sprint Review is timeboxed to a maximum of four hours for a one-month Sprint. For shorter Sprints, the event is usually shorter." (Schwaber and Sutherland, 2020, p. 9, para. 10)
R13	"The Sprint Review is the second to last event of the Sprint." (Schwaber and Sutherland, 2020, p. 9, para. 10)
R14	"The Sprint Retrospective concludes the Sprint." (Schwaber and Sutherland, 2020, p. 10, para. 4)

Table 8 continued from previous page

R15	"The Sprint Retrospective is timeboxed to a maximum of three hours for a one-month Sprint. For shorter Sprints, the event is usually shorter." (Schwaber and Sutherland, 2020, p. 10, para. 4)
R16	"For each selected Product Backlog item, the Developers plan the work to create an Increment that meets the Definition of Done." (Schwaber and Sutherland, 2020, p. 8, para. 9)
R17	"The Sprint Goal, the Product Backlog items selected for the Sprint, plus the plan for delivering them are together referred to as the Sprint Backlog." (Schwaber and Sutherland, 2020, p. 9, para. 1)
R18	"Multiple Incrementes may be created within a Sprint." (Schwaber and Sutherland, 2020, p. 12, para. 1)
R19	"The moment a Product Backlog item meets the Definition of Done, an Increment is born." (Schwaber and Sutherland, 2020, p. 12, para. 4)
R20	"If the Definition of Done for an increment is part of the standards of the organization, all Scrum Teams must follow it as a minimum." (Schwaber and Sutherland, 2020, p. 12, para. 6)
R21	"If there are multiple Scrum Teams working together on a product, they must mutually define and comply with the same Definition of Done." (Schwaber and Sutherland, 2020, p. 12, para. 7)
R22	"The Sprint Review should never be considered a gate to releasing value." (Schwaber and Sutherland, 2020, p. 12, para. 1)
R23	"The Product Owner is one person, not a committee." (Schwaber and Sutherland, 2020, p. 6, para. 4)
R24	"The Product Owner is also accountable for effective Product Backlog management, which includes: Developing and explicitly communicating the Product Goal; Creating and clearly communicating Product Backlog items; Ordering Product Backlog items; Ensuring that the Product Backlog is transparent, visible and understood." (Schwaber and Sutherland, 2020, p. 6, para. 1)

Table 8 continued from previous page

R25	"If the Product Owner or Scrum Master are actively working on items in the Sprint Backlog, they paraticipate as Developers." (Schwaber and Sutherland, 2020, p. 9, para. 4)
R26	"Only the Product Owner has the authority to cancel the Sprint." (Schwaber and Sutherland, 2020, p. 8, para. 3)
Proposed Rules Below:	
R27	Development teams consist of approximately 8 developers. Based on R2 and R3.
R28	The development lifecycle must not include unusual workflow or changes in the issue statuses. For instance, an issue marked as 'complete' should not re-appear with a status of 'unresolved' or 'open' after it has been completed.
R29	The Scrum Team does not include any other people (e.g. a manager who doesn't do tasks).
R30	Issues resolved in previous increments do not appear again in future increments.
R31	Each Scrum Sprints can be considered a short project, therefore they are uniquely identified.
R32	Issue have a corresponding type (bug, task, etc.).
R33	Each issue belongs to a specific Sprint.
R34	All issues must be uniquely identifiable.
R35	Scrum Sprints have a starting time and date.
R36	Scrum Sprints have a completion time and date.
R37	The number of PBIs selected from the Backlog for a Sprint depends on the developers, but it should not be zero PBIs per sprint.

Table 8 continued from previous page

R38, R39	"The Product Owner is also accountable for effective Product Backlog management, which includes: Developing and explicitly communicating the Product Goal; Creating and clearly communicating Product Backlog items; Ordering Product Backlog items; Ensuring that the Product Backlog is transparent, visible and understood." (Schwaber and Sutherland, 2020, p. 5, para. 1) "Product Backlog refinement is the act of breaking down and further defining Product Backlog items into smaller more precise items. This is an ongoing activity to add details, such as a description, order, and size. Attributes often vary with the domain of work." (Schwaber and Sutherland, 2020, p. 10, para. 9)

Table 9. Data fields Used to Map to Scrum Rules

Rule ID	**Data Fields Used for Scrum Rules Mapping**
R1	sprint
R2	N/A
R3	N/A
R4	sprint
R5	sprint
R6	project, sprint, key
R7	sprint
R8	sprint, key, assignee.name, created, resolutiondate
R9	N/A
R10	N/A
R11	N/A
R12	N/A
R13	N/A
R14	N/A

Table 9 continued from previous page

R15	N/A
R16	N/A
R17	N/A
R18	N/A
R19	N/A
R20	N/A
R21	N/A
R22	N/A
R23	N/A
R24	project, sprint, key, resolutiondate, description, issuetype.name, priority.name, summary
R25	N/A
R26	N/A
R27	sprint, assignee.name
R28	sprint, key, status.name, status_id, statusCategory_name
R29	N/A
R30	sprint, key, created, resolutiondate
R31	sprint
R32	sprint, key, issuetype.names
R33	sprint, key
R34	sprint, key
R35	sprint
R36	sprint
R37	sprint, key
R38	sprint, key, created
R39	sprint, key, resolutiondate

Table 10. Total number of verifiable and non-verifiable rules

	Number of rules we can verify through the data fields	Number of rules we cannot verify through the data fields	Total Number of Rules
	19	20	39
Total %	49%	51%	

Table 11. Useful Data Fields

Rank	**Data Field**	**Number of verifiable rules where data field is useful**	**Percentage**
1	sprint	19	100%
2	key	11	57.89%
3	created	4	21.05%
4	resolutiondate	4	21.05%
5	assignee.name	2	10.52%
6	project	2	10.52%
7	issuetype.name	2	10.52%
8	priority.name	1	5.26%
9	description	1	5.26%
10	summary	1	5.26%
11	status.name	1	5.26%
12	status_id	1	5.26%
13	statusCategory_name	1	5.26%

Table 12. Non-verifiable Scrum Rules due to Insufficient Data for Scrum Events and Roles

Rule ID	**Scrum Event**	**Scrum Role**
R2, R3	Sprint	Scrum Team
R9	Sprint Planning	Developers
R10, R11	Daily Scrum	Scrum Team

Table 12 continued from previous page

R12, R13	Sprint Review	Scrum Team
R14, R15	Sprint Retrospective	Scrum Team
R16, R17	Sprint Planning	Developers
R18	Product Backlog	Developers
R19	Increment, DoD	Developers
R20	Sprint, Increment	Scrum Team
R21	Sprint, Increment, DoD	Scrum Team
R22	Sprint Review, Increment	Developers, PO
R23	Sprint	PO
R25	Daily Scrum	PO, SM
R26	Sprint	PO
R29	Sprint	Scrum Team

5.2 Research Question 2

In order to answer the - *RQ2: "What are the most common Scrum practices among the development teams from the open-source projects?"*, each of the verifiable rules has been checked through a script of code written in Python programming language[9]. The output for each rule is a binary result, meaning that the Scrum rule either passes (1) or fails (0) for the given project. For non-verifiable rules, the result is reported as non-applicable (N/A). In Table 14, these rules are placed at the end of the table, starting from the rank 22 until the end of the table. The calculation is done through the formula shown in the Methodology subsection 4.2, where the number of projects where the particular rule has passed is divided by total number of verifiable rules in the set. Afterwards, in Table 14, the Scrum practices are ranked, starting from the practices that have shown higher compliance, i.e. most popular/common practices towards the less popular practices among the teams. One important thing to note, is that from the set of

[9]The coding scripts used to programmatically implement the Scrum rules can be found via the following URL: *https://github.com/mirlindm/MasterThesis-Mirlind-2021/tree/main/Notebooks%20-%20Coding%20Scripts%20*

identified rules, there are 5 rules related to the duration or time of Scrum sprints. These rules are:

- R1: No more than five weeks should elapse for a single sprint.
- R4: The duration of all sprints should follow similar pace.
- R5: The next Sprint execution should begin only after the previous Sprint's resolution.
- R7: No considerable amount of time should elapse between the finish of a sprint and the beginning of the new sprint.
- R35: There should be timestamp indicating the sprints kick-off.
- R36: There should be timestamp indicating the sprints completion.

The aforesaid rules' calculation require data indicating sprint's start and completion time and date, however, in the case of the studied projects, this information is not present in most projects' data. Table 13 presents sample values present in the sprint data field for each project.

Table 13. Sample Values of Sprint Data Field

Project	**Sprint Data Field Sample Value**
XD	Sprint 68
APSTUD	2012 Sprint 15
TISTUD	2012 Sprint 18
MOBILE	"[’com.atlassian.greenhopper.service.sprint.Sprint@245be8a [id=289,rapidViewId=43,state=ACTIVE, name=Moodle App 3.8.2, startDate=2020-03-30T11:10:08.305Z, endDate=2020-04-03T16:56:00.000Z, completeDate=<null>, activatedDate=2020-03-30T11:10:08.305Z, sequence=289,goal=]’]"

Table 13 continued from previous page

MDL	"[’com.atlassian.greenhopper.service.sprint.Sprint@40c6b8aa [id=237,rapidViewId=126,state=ACTIVE, name=Moodle 3.9 - Moppies Kanban, startDate=2019-09-15T13:49:00.000Z, endDate=2019-10-31T14:49:00.000Z, completeDate=<null>,sequence=237,goal=<null>]’]"
DNN	Sprint 2; Content 8.1.0 - Sprint 3;
MESOS	Mesosphere Sprint 34
MULE	startDate=2016-04-27T11:27:32.496-05:00
NEXUS	Sprint 68 - Föhn
TIMOB	Documentation Sprint 03 - 2016; Sprint 2012-15 API

Table 14. Most Popular Scrum Practices

Rule ID	**Number of Projects where Rule Passed (%)**	**Number of Projects where Rule Could Be Computed (%)**	**Final Result (%)**
R32	10	10	100%
R6	10	10	100%
R37	10	10	100%
R38	10	10	100%
R30	9	9	100%
R34	9	10	90%
R31	8	9	80%
R35	2	3	67%
R36	2	3	67%
R24	6	10	60%
R27	5	10	50%
R28	3	9	33%
R5	1	3	33%
R8	1	10	10%

Table 14 continued from previous page

R1	0	3	0%
R4	0	3	0%
R7	0	3	0%
R33	0	10	0%
R39	0	10	0%

5.3 Research Question 3

In order to answer the - *RQ3: "What are the compliance levels of the development teams towards the implemented Scrum practices?"*, the number of rules that pass for a given project, divided by the total number of rules is calculated, which provides insights on how compliant were the teams towards Scrum practices. These results can be seen in Figure 15, and are further discussed in the succeeding section [10].

Table 15. Projects' Compliance to Scrum Rules

Project Name	**Total Passing Rules**	**Total Computable Rules**	**Final Result (%)**
XD	10	13	77%
APSTUD	9	13	69%
TIMOB	9	13	69%
MESOS	9	13	69%
DNN	8	13	62%
NEXUS	7	13	62%
TISTUD	8	13	62%
MOBILE	11	19	58%
MULE	8	19	47%
MDL	7	17	41%

[10]The detailed results for RQ2 and RQ3 can be found via the following URL: *https://github.com/mirlindm/MasterThesis-Mirlind-2021/tree/main/RQ2-RQ3%20Results*

5.4 Study Validation

This section presents the validation of this research. In order to verify the validity of the results derived from this study, the approach of ensuring functional correctness of the implemented rules through software testing strategies has been followed. Using the coverage tool *Coverage.py*[11], the tests written for each rule have been monitored and a report of their effectiveness has been compiled. The testing scripts and documentation of the test cases can be found in its respective directory on GitHub[12]. Through 46 unit tests written for 19 computable rules, most scenarios pertaining to the various scenarios that occur have been tested and properly documented. Table 16 below shows the test coverage report for all of the computed rules. In addition, a snippet of a unit test for a particular rules is proved in Appendix II through Figure 10.

Table 16. Unit Tests Coverage

Rule ID	**Test IDs**	**Test Suite**	**Statements**	**Missed**	**Coverage**
R1	T1, T2, T3	test_R1.py	41	6	85%
R4	T1, T2	test_R4.py	36	6	83%
R5	T1, T2, T3	test_R5.py	53	1	98%
R6	T1, T2	test_R6.py	17	0	100%
R7	T1, T2	test_R7.py	41	1	98%
R8	T1, T2	test_R8.py	48	1	98%
R24	T1, T2, T3, T4, T5, T6, T7,	test_R24.py	53	0	100%
R27	T1, T2, T3	test_R27.py	28	0	100%
R28	T1, T2	test_R28.py	37	1	97%
R30	T1, T2	test_R30.py	47	2	96%
R31	T1, T2	test_R31.py	17	0	100%

[11] https://coverage.readthedocs.io/en/coverage-5.5/

[12] *URL to Unit Tests Notebook Implementation and Test Cases Documentation: https://github.com/mirlindm/MasterThesis-Mirlind-2021/tree/main/Study%20Validation*

Table 16 continued from previous page

R32	T1, T2	test_R32.py	24	5	79%
R33	T1, T2	test_R33.py	24	5	79%
R34	T1, T2	test_R34.py	17	0	100%
R35	T1, T2	test_R35.py	21	0	100%
R36	T1, T2	test_R36.py	24	2	92%
R37	T1, T2	test_R37.py	19	0	100%
R38	T1, T2	test_R38.py	29	6	79%
R39	T1, T2	test_R39.py	26	6	77%
TOTAL			**602**	**42**	**93%**

6 Discussion

6.1 Scrum Rules Verification Using Data from Jira

Following the results for RQ1: *"To what extent are data extracted from Jira useful to apply conformance checking techniques?"*, tables 7, 8, 9 show that we can use JIRA data to verify roughly half of the identified Scrum rules or practices. Out of 39 Scrum rules identified, 19 rules can be mapped and verified with the available data, hence rendering the data to be useful to moderate extent. Anyhow, there are 20 remaining rules that unsuccessfully, cannot be checked through the data we have available.

From Table 10, we conclude that during this research, we could check roughly half of the Scrum rules using the available Jira data from the open-source projects. However, for a big amount of rules we do not have sufficient data to perform conformance checking against. Out of these rules, there are 4 rules (R2, R3, R9, R29) that can only be partially verified through the data (only initial part of the rules can be computed/verified), however, it will still be insufficient to check for full Scrum compliance. Hence, these are considered as rules that cannot be verified.

Table 11 provides insights on the usefulness of the data fields, needed to check the Scrum rules programmatically. The *sprint* data field has been used in checking all of the rules that are possible to check, which renders this field to be the most useful for this study.

Moreover, the data field which indicates the issues, i.e. the *key* field, is also used in 11 out of 19 verifiable rules. Considering that in software development, time is a critical factor, and that all of the Scrum Events are time-bound, other useful data fields for this study are the ones that signify the start and end date and time for sprints and issues. The fields *created* and *resolutiondate* represent the start and end time of an issue, respectively. We do not have distinct fields indicating the starting and completion time of the actual sprints, however, the data field *sprint* itself, for several projects, carries valuable information about each sprint, including the start and end time. This is another reason behind the high usefulness of this attribute in the data.

By observing the rules that cannot be fully verified using the data, some patterns in the data input as regards the Scrum Events and Roles, were discovered where the data is not enough to perform (full) Scrum Compliance, as shown in Table 12. As we can observe from the table, the events for which we have more rules that can't be checked are the Sprint, followed by Daily Scrum, Sprint Review and Sprint Retrospective. On the contrary, the Scrum Roles for which there are more rules with insufficient data, are: Developers, Product Owner, Scrum Master, respectively. This signifies that we are unable to verify the number of Product Owners or Scrum Masters in the Scrum Team, and if these roles acted correctly upon their accountabilities.

Proceeding with the study results of the RQ1, with 49% of the identified Scrum rules and framework being covered by the data, we can conclude that data taken from Jira is useful to be applied for conformance checking of the agile framework of Scrum to nearly half of the defined Scrum practices. In order to achieve higher coverage of the framework, we greater amount of useful data to be captured by the issue tracking systems, in this case Jira. Finally, since it is not of our knowledge that a similar research has been conducted before, the answer to this research question furthers our understanding on the type of data that is necessary and useful for applying conformance checking onto agile software development frameworks, in our case Scrum, and hopefully unlocks new research ideas within the similar ripple to be carried out in the future.

6.2 Popular Scrum Practices among the Software Projects

According to the results for RQ2: *"What are the most common Scrum practices among the development teams from the open-source projects?"*, the most frequent or popular

Scrum practices among the studied projects are shown in Table 14. These practices relate to the rules that are possible to check through the obtainable data. The rules that were not impossible to verify are not placed in the table.

There are 11 rules whose final result depends on the availability of information about the sprints. In Table 14, these rules are: R1, R4, R5, R6, R7, R8, R27, R31, R35, R36, R37. These rules check various characteristics or attributes of sprints, such as completion (duration) time, time spent transitioning between two sprints, order of sprints time-wise. However, as we can see from Table 13, there are only 3 projects for which we can apply the R1, R4, R5, R7, R35, R37 rules. These projects are: MOBILE, MDL and MULE. Unfortunately, we do not have complementary data about the sprint data field in most of the projects, such as start date and end date. Reasons behind it might include that the teams were not logging this information in Jira, or they did not keep track of the sprints' start and end dates. The latter one highly opposes Scrum Guide's principles. Anyhow, out of the projects for which we have enough data to check these rules, results showed that MULE was more compliant to these particular rules as compared to the other projects. Again, potential reasons that lead to such results lie on the data themselves. While carrying out the analysis, it was easily noticeable that the data recording in Jira stemming from the development activities, has been the most seamless one for MULE, as compared to the other projects (the data recorded contains clearer information about consecutive sprints, which is not the case with the other two projects, in which, the information regarding sprints is not sufficiently consistent to verify the sprint-related rules).

Among the rules that target the timeline of sprints (duration, gap between sprints, overlapping sprints, etc.), results show that although the 3 projects followed the formal approach overall, most of the practices do not occur as prescribed. For instance, in projects like MDL (95%) and MULE (99%), most of the sprints took place within less than 4 weeks, as expected by the Scrum Guide, however, only a small portion of sprints from MOBILE (27%) adhered to this specification. Nevertheless, the result obtained from MOBILE project could potentially derive from an internally adapted version of Scrum, which incorporates sprints longer than 4 weeks.

On the contrary, as regards overlapping sprints, results showcase that MOBILE and MULE have highly complied to this rule with 95% and 100% of sprints following non-concurrent timeline, meaning that the next sprints have mostly began after the previous

sprint has ended. However, MDL project showed that only 30% of its sprints adhered to this practice.

On the other hand, when it comes to the rules that were checked through the total number of issues, all 10 projects were analyzed. These rules are mostly used to check the attributes pertaining to the issue reports, such as time elapsed on completing the issues, idle time from when an issue was resolved until the next one begins, checking whether issues have the relevant and meaningful details for the developers, Scrum Team and the agile process itself. Through these rules, we can also obtain clearer insights on what Scrum practices were followed more by the teams and which ones result in low popularity among developers. As we can see from the results, only two projects used story points in their development process, rendering Scrum practices related to story points (user story splitting, etc.) to be low in terms of popularity and use. There are practices that resulted in high compliance and usage, such as keeping the development team nimble and small, volunteering for new tasks when older ones get completed, maintaining well-documented issue reports, and more. Some of these rules are also influenced by Jira default configuration to some extent, alluding to the fact that the system makes it mandatory for issues to contain a valid name/ID, having a type assigned to each issue, etc.

One thing noted is that in all projects, not all issues were allocated to sprints. For instance, the MDL project contained 63273 issues in total, out of which, 60540 issues (96%) were not identified as part of any of the 269 sprints conducted for the development process. Reasons behind it may be that sprints were defined only for a short period of the project. Nonetheless, this pattern of issues not being allocated among the sprints was common among all open-source projects, however, the amount of issues belonging to no sprints in other projects was relatively smaller than the aforementioned project.

Another observation worth investigating more, points out to the fact that there are a few issues in two projects that did not have a reporter name, which goes against the Jira software defaults, in which the reporter represents the person who created the issue. These cases are present in MDL (3 issues only or 0.0047%) and NEXUS (3 issues or 0.3%).

Two other common patterns were spotted across the projects. First one relates to high amounts of incomplete issues present in the respective backlog of each team. According to the Scrum Guide: "All incomplete Product Backlog Items are re-estimated and put

back on the Product Backlog. The work done on them depreciates quickly and must be frequently re-estimated." [7], the incomplete issues of a certain sprint transition into the backlog of another sprint, where developers work (again) towards their completion. Considering this statement and the implemented analysis, none of the projects fully complied to this particular Scrum practice, thus rendering it to be one of the least popular practices across the studied projects. The other pattern identified in the data targets the developers. There are many issues that were never assigned to any of the active developers, nor even the contributors of the team, however they are marked as 'done' or 'in-progress'. Consequently, this could be one reason for the incomplete issues (developers starting to work on the issues for some time and leaving them unattended after a while, while unassinging themselves from the issue reports), which again, does not align with the principles and guidelines listed in the Scrum Guide, which states that Scrum teams are self-organizing and that developers self-assign tasks and swiftly volunteer to work on other tasks, once finished with current ones. The findings of this research converge with the results from [21], in the manner that Process Mining techniques and capabilities can be successfully applied onto agile development models and can benefit the software development teams, process and product outcome. The way it could deliver value to the development teams is by providing reports comprising of deviations towards the established process, which can be used to increase compliance by mapping out the practices that have been violated and proposing ways of increasing the adherence to these particular practices.

The results of RQ2 harmoniously align with the conclusions from the study done in [28]. Although the other study combines a tool with defined conformance metrics for multiple agile models, they too analyze development artifacts (from GitHub) in order to detect violations of agile practices. Both studies agree that there are areas of improvement when adopting Scrum, and both results sustain these statements.

Moreover, the RQ2 results also line up with the findings presented in [27], in which conformance checking practices have been successfully used in combination with machine learning to discover patterns in developers' behavior and code to compare it against the prescribed processes.

Nonetheless, what makes this study distinctive from alike endeavors, is having data as the pivotal point and focusing solely on Scrum by utilizing official Scrum documentation to support rule-based conformance checking against real data from 10 open-source projects.

6.3 Software Projects' Scrum Compliance

Taking into consideration the results for *"RQ3: What are the compliance levels of the development teams towards the implemented Scrum practices?"*, table 15 reveals the compliance that the software teams have shown against the Scrum rules employed in this thesis. Overall, with 19 computable rules, i.e. 49% of the framework, the compliance of Scrum recorded among the projects ranges from over 30% to less than 80%, depending on the project and the computable rules per each project.

From Table 13 we can notice that for projects *MOBILE*, *MDL* and *MULE*, the available data conveys information about the start date and end date of the sprints. Whereas, for the other 7 projects, we do not possess enough data for the sprints' timelines, therefore the 6 rules listed in the *Results* section of RQ2, are unable to be calculated for these projects. In order to accomplish more accurate and thorough results, including the rules that are not possible to check with our datasets, we need more data and most importantly, the data at our disposal needs to be of high quality. In that manner, the results shown in both Table 14 and Table 15 could present an even more holistic reality of the development activities within the teams.

As we can notice from the Table 15, the teams that showcased higher compliance to the framework, i.e. the teams for which, higher number of Scrum rules resulted to be true for their respective amount of issues and sprints, is *XD*, followed by *APSTUD*, *TIMOB*, and *MESOS*. On the contrary, the teams that have recorded the least compliance (with the lower number of Scrum rules passing) are *MOBILE, MULE* and *MDL*.These projects are the ones with more issues. This could be another reason behind the low compliance. The more issues you have, the higher the probability of failing the rules is. This is because there are more chances of finding non-valid issues that will spoil the entire rule calculation since the rule verification yields either 0 or 1.

Generally, the data recording (data logging in the issue tracker) from the development process for *XD* project seems to have been more successful, as can be inferred by the results and data quality. Sprints and issues are well documented, there are less data discrepancies, and the datasets for these projects are overall more consistent. Unfortunately, the datasets for the remaining projects are less qualitative, comprising of less consistent data, which greatly hindered the analysis, rendering the results to display lower compliance of the projects against the verifiable Scrum practices.

The results of this research question align with the results showed in [5], in which the studied team showed high compliance to the company's internal Scrum adapted version. Moreover, the findings from this thesis line up with the findings from [23], in which Scrum framework has shown high compliance against the ISO standard it was being compared to.

6.4 Limitations

Proceeding with the results from the study design and framework chosen for it, it is important to acknowledge limitations pertaining to the data. In order to achieve more accurate results, additional data from external sources are needed, which convey relevant information about the various events, roles and artifacts that are vital to the Scrum framework. On the contrary, we are unable to check for several Scrum events and roles, due to of the lack of data for these specific components. A lot of invaluable information cannot be checked, as a consequence of it not being captured by Jira. For instance, Daily Scrum is a crucial event taking place whilst Sprints, which with the current data, we are unable to check it through any of rules affecting it. Also, the accountabilities of the Scrum Master and Product Owner are not seized by the data, hence, we cannot be sure whether these two roles act upon their duties, accordingly.
Furthermore, following the limitations stated above, this research study carries some validity threats to be addressed.
An important validity aspect concerns the *conclusions* that we can draw from the results. The chain of consecutive activities implemented to reach the final results carry on risks related to the data cleaning and processing steps. Many issues from each project's dataset were removed due to duplicate or repetitive values. Then, for programmatically implementing and checking certain rules using the data fields, other data processing steps were undertaken in-place, which resulted in more issues being removed from particular project's data. For example, *R27* checks for number of active developers within each sprint, and its calculation required to remove all other non-active developers also known as contributors to the open-source projects, which as a consequence, removed all the issues that were completed by them. Another case was whilst checking the rules regarding the timeline of sprints, in which, the issues that did not belong to any of the sprints, had to be discarded for the analysis of these particular rules. These instances

could influence the final results, therefore the derived conclusions as well. As regards the Scrum rules extracted from Scrum Guide, there might be more rules to be compiled than the ones defined in this study. Additionally, for future work, we can include other relevant Scrum sources to increase the number of rules, in order to have a higher Scrum coverage.
Another major aspect encompasses the extent to which we can generalize the findings of this study. Since the data analyzed belongs to real life open-source projects, the derived insights do not represent all other open-source projects within their community. As already mentioned previously, the team structure is variable from one project to the other, the technologies and tools vary too. Moreover, some projects clearly did not show evidence to have followed certain agile practices (story points, sprints, etc.) in comparison to others. Therefore, complementary research is required in order to validate the results that this study yielded.

7 Conclusions and Future Work

Checking the conformance of companies' or teams' real development process against the prescribed standard or model adopted, is an important activity that adds up to the teams agility and delivers useful insights on quality enhancement. Increasingly, software companies conduct auditing activities targeting quality and compliance issues. In this study, the data of ten open-source projects extracted from Jira have been used in order to apply rule-based conformance checking against the Scrum agile framework. Simultaneously, a set of Scrum rules was meticulously extracted from the Scrum Guide [7], where each rule relates to a Scrum practice or represents one. As the collected data and defined Scrum rules are the basis of this study, their quality and punctilious extraction are predominantly responsible for the quality of overall results. These rules have been used for conformance checking, in order to identify whether the development teams were complying to the guidelines of Scrum, and to what extent they were doing so. Moreover, this study highlights the most popular Scrum practices among the projects that were analyzed, and compares them in order to identify the teams that were adhering to the Scrum framework more than the others. This way, we can find patterns and insights that can be proposed to other teams to increase their Scrum adoption and efficiency.
The obtained results showed that not all Scrum practices can be verified through the data

extracted from Jira. Although a great number of the defined Scrum rules were programmatically checked, there is still a big subset of the rules that was not possible to check. Scrum practices related to the Product Owner's and Scrum Master's responsibilities within the Scrum Team could not be checked, due to the lack of data related to these Scrum roles. Furthermore, similar limitations persist for Scrum Events, such as Daily Scrum, Scrum Review, and Retrospective.

In addition, the most popular Scrum practices among the open-source projects involve Scrum development team composition, practices related to the sprint timeline, practices related to product backlog and sprint backlog items, well-maintained issue documentation throughout the lifecycle and workflow of issues, etc. On the other hand, some projects showed higher compliance than others. For example, XD team was one of the most compliant teams against the analyzed rules. Whereas projects like MDL, MULE, and MOBILE has resulted in the lower compliance to the Scrum framework. However, acknowledging the presence of several limitations faced, we are unable to surely state how differently or similarly the results of the research questions tackled by this thesis would be, in case more qualitative data targeting the non-verifiable rules were available. As regards the complementary work to be done henceforth, a full validation of the results is needed. This will be done by distributing a questionnaire to the respective projects and use the responses to determine the validity of the obtained results.

Ideally, complementing the study with model-based conformance checking is another research objective to tackle, as per future attempts. In this case, the business model of the development process can be employed for model-based conformance checking, in which it can be compared against the discovered process model from the event logs. For instance, obtaining the process model of any of the studied projects can be used to compare to the process model of Scrum depicted in Figure 4 and Figure 5. In addition, we can take advantage of the Conformance Checking techniques presented in Table 1, in order to obtain a more accurate picture of the deviations or violations concerning to the practices of the framework.

Moreover, for future endeavors, it will also be valuable to agile teams to be supported with a forward-compliance checking approach, incorporating machine learning models or predictive capabilities, that will report to the Scrum Teams after each completed Sprint event in cases the Scrum practices are being violated or the team is deviating in any of them. This will enable the entire Scrum Team to remain within the Scrum

boundaries and increase their adherence levels towards the framework. Finally, the more data and most significantly, with more elevated quality in them, that can be captured from issue-tracking tools like Jira, the more accurate the picture of the entire actual agile development process will be, and the better could we perform conformance checking in the agile software development realm, which as a consequence, will lead to more informative insights to the agile software teams.

References

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

[2] S. Zaouali and S. A. Ghannouchi, "Proposition of an approach based on bpm to manage agile development processes," in *2016 Third International Conference on Systems of Collaboration (SysCo)*, pp. 1–6, 2016.

[3] G. S. Matharu, A. Mishra, H. Singh, and P. Upadhyay, "Empirical study of agile software development methodologies: A comparative analysis," *SIGSOFT Softw. Eng. Notes*, vol. 40, pp. 1–6, Feb. 2015.

[4] E. Scott, G. Rodríguez, Álvaro Soria, and M. Campo, "Are learning styles useful indicators to discover how students use scrum for the first time?," *Computers in Human Behavior*, vol. 36, pp. 56 – 64, 2014.

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

[6] T. Jaturogpattana, P. Arpasat, K. Kungcharoen, S. Intarasema, and W. Premchaiswadi, "Conformance analysis of outpatient data using process mining technique," in *2017 15th International Conference on ICT and Knowledge Engineering (ICT KE)*, pp. 1–6, 2017.

[7] K. Schwaber and J. Sutherland, *The Scrum Guide*, pp. 133–152. John Wiley & Sons, Ltd, 2012.

[8] K. Schwaber and J. Sutherland, "Der scrum guide," *tobechanged*, 2014.

[9] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, "Agile software development methods: Review and analysis," *Proc. Espoo 2002*, pp. 3–107, 01 2002.

[10] M. Stoica, M. Mircea, and B. Ghilic-Micu, "Software development: Agile vs. traditional," *Informatica Economica*, vol. 17, pp. 64–76, 12 2013.

[11] K. Schwaber, *Agile Project Management with Scrum*. Redmond, WA: Microsoft Press, 2004.

[12] M. R. M. Chopade and N. S. Dhavase, "Agile software development: Positive and negative user stories," in *2017 2nd International Conference for Convergence in Technology (I2CT)*, pp. 297–299, 2017.

[13] W. Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, vol. 136. Springer, 01 2011.

[14] M. Dumas, M. La Rosa, J. Mendling, and H. Reijers, *Fundamentals of Business Process Management*. Springer Berlin Heidelberg, 03 2018.

[15] W. van der Aalst, A. Adriansyah, A. K. A. de Medeiros, F. Arcieri, T. Baier, T. Blickle, J. C. Bose, P. van den Brand, R. Brandtjen, J. Buijs, A. Burattin, J. Carmona, M. Castellanos, J. Claes, J. Cook, N. Costantini, F. Curbera, E. Damiani, M. de Leoni, P. Delias, B. F. van Dongen, M. Dumas, S. Dustdar, D. Fahland, D. R. Ferreira, W. Gaaloul, F. van Geffen, S. Goel, C. Günther, A. Guzzo, P. Harmon, A. ter Hofstede, J. Hoogland, J. E. Ingvaldsen, K. Kato, R. Kuhn, A. Kumar, M. La Rosa, F. Maggi, D. Malerba, R. S. Mans, A. Manuel, M. McCreesh, P. Mello, J. Mendling, M. Montali, H. R. Motahari-Nezhad, M. zur Muehlen, J. Munoz-Gama, L. Pontieri, J. Ribeiro, A. Rozinat, H. Seguel Pérez, R. Seguel Pérez, M. Sepúlveda, J. Sinur, P. Soffer, M. Song, A. Sperduti, G. Stilo, C. Stoel, K. Swenson, M. Talamo, W. Tan, C. Turner, J. Vanthienen, G. Varvaressos, E. Verbeek, M. Verdonk, R. Vigo, J. Wang, B. Weber, M. Weidlich, T. Weijters, L. Wen, M. Westergaard, and M. Wynn, "Process mining manifesto," in *Business Process Management Workshops* (F. Daniel, K. Barkaoui, and S. Dustdar, eds.), (Berlin, Heidelberg), pp. 169–194, Springer Berlin Heidelberg, 2012.

[16] M. R. Peña and S. Bayona-Oré, "Process mining and automatic process discovery," in *2018 7th International Conference On Software Process Improvement (CIMPS)*, pp. 41–46, 2018.

[17] M. A. Ghazal, O. Ibrahim, and M. A. Salama, "Educational process mining: A systematic literature review," in *2017 European Conference on Electrical Engineering and Computer Science (EECS)*, pp. 198–203, 2017.

[18] C. Romero, R. Cerezo, A. Bogarín, and M. Sánchez-Santillán, *EDUCATIONAL PROCESS MINING: Applications in Educational Research*, pp. 1–28. Data Mining and Learning Analytics, 09 2016.

[19] X. Lu, D. Fahland, and W. Aalst, "Conformance checking based on partially ordered event data," in *Business Process Management Workshops*, vol. 202, pp. 75–88, 01 2015.

[20] E. Batista and A. Solanas, "Process mining in healthcare: A systematic review," in *2018 9th International Conference on Information, Intelligence, Systems and Applications (IISA)*, pp. 1–6, 2018.

[21] V. Rubin, I. Lomazova, and W. Aalst, "Agile development with software process mining," in *2014 International Conference on Software and System Process (ICCSP 2014, Nanjing, China, May 26-28, 2014)"*, 05 2014.

[22] M. Adel and A. Bahaa, "The discovery of the implemented software engineering process using process mining techniques," *International Journal of Advanced Computer Science and Applications*, vol. 7, 02 2016.

[23] S. Galvan, M. Mora, R. V. O'Connor, F. Acosta, and F. Alvarez, "A compliance analysis of agile methodologies with the iso/iec 29110 project management process," *Procedia Computer Science*, vol. 64, pp. 188 – 195, 2015. Conference on ENTERprise Information Systems/International Conference on Project MANagement/Conference on Health and Social Care Information Systems and Technologies, CENTERIS/ProjMAN / HCist 2015 October 7-9, 2015.

[24] M. Kharbili, S. Sebastian, and W. Aalst, "Business process compliance checking: Current state and future challenges.," in *Modellierung betrieblicher Informationssysteme (MobIS 2008)*, pp. 107–113, 01 2008.

[25] F. Moyon, K. Beckers, S. Klepper, P. Lachberger, and B. Brügge, "Towards continuous security compliance in agile software development at scale," *2018 IEEE/ACM 4th International Workshop on Rapid Continuous Software Engineering (RCoSE)*, pp. 31–34, 2018.

[26] B. Gallina, F. UL Muram, and J. Castellanos Ardila, "Compliance of agilized (software) development processes with safety standards: a vision," in *4th international workshop on Agile Development of Safety-Critical Software*, pp. 1–6, 05 2018.

[27] J. Caldeira and F. B. e. Abreu, "Software development process mining: Discovery, conformance checking and enhancement," in *2016 10th International Conference on the Quality of Information and Communications Technology (QUATIC)*, pp. 254–259, 2016.

[28] C. Matthies, T. Kowark, K. Richly, M. Uflacker, and H. Plattner, "Scrumlint: Identifying violations of agile practices using development artifacts," in *Proceedings of the 9th International Workshop on Cooperative and Human Aspects of Software Engineering*, pp. 40–43, 05 2016.

[29] E. Scott, D. Pfahl, R. Hebig, R. Heldal, and E. Knauss, "Initial results of the helena survey conducted in estonia with comparison to results from sweden and worldwide," in *Product-Focused Software Process Improvement* (M. Felderer, D. Méndez Fernández, B. Turhan, M. Kalinowski, F. Sarro, and D. Winkler, eds.), (Cham), pp. 404–412, Springer International Publishing, 2017.

[30] E. Scott and D. Pfahl, "Using developers' features to estimate story points," in *Proceedings of the 2018 International Conference on Software and System Process*, ICSSP '18, (New York, NY, USA), pp. 106–110, Association for Computing Machinery, 2018.

[31] E. Scott, K. Charkie, and D. Pfahl, "Productivity, turnover, and team stability of agile teams in open-source software projects," in *Proceedings of the 2018 International Conference on Software and System Process*, pp. 124–131, 08 2020.

Appendices

I. File Names, Descritpions and GitHub URLs

Table 17. File Names, Descritpions and GitHub URLs (under the respective file)

File Name	Description
RQ1 - Scrum Rules Table.pdf	Scrum Rules file containing the set of defined Scrum Rules with all necessary information.
https://github.com/mirlindm/MasterThesis-Mirlind-2021/blob/main/RQ1%20Results/RQ1_%20Scrum%20Rules%20Table.pdf	
RQ2-RQ3 Results.pdf	Results for RQ2 and RQ3 presented in a matrix, where 10 projects are listed as columns and the rows represent the 39 defined Scrum Rules. Additional information on the results (binary output as well as detailed results on the number of issues or sprints that pass or fail for each rule)
https://github.com/mirlindm/MasterThesis-Mirlind-2021/blob/main/RQ2-RQ3%20Results/RQ2-RQ3%20Results.pdf	
Detailed Results for RQ2-RQ3 .pdf	Very detailed results for each rule and project, providing all information about the execution and processing of the rules, as well as the results.
https://github.com/mirlindm/MasterThesis-Mirlind-2021/blob/main/RQ2-RQ3%20Results/RQ2%20-%20Detailed%20Results.pdf	
Scrum Rules Implementation.ipynb	Python Notebook containing all the coding scripts that were implemented in order to programmatically verify the rules.
https://github.com/mirlindm/MasterThesis-Mirlind-2021/blob/main/Notebooks%20-%20Coding%20Scripts%20/Scrum%20Rules%20Implementation.ipynb	
Data Cleaning Notebook.ipynb	Python Notebook used initially for data cleaning and separating project's data to individual datasets.
https://github.com/mirlindm/MasterThesis-Mirlind-2021/blob/main/Notebooks%20-%20Coding%20Scripts%20/Data%20Cleaning%20Notebook.ipynb	
Testing Scrum Rules.ipynb	Python Notebook containing all the unit tests written for each rule. Testing library used is Pytest.
https://github.com/mirlindm/MasterThesis-Mirlind-2021/blob/main/Study%20Validation/Testing%20Scrum%20Rules.ipynb	

Table 17 continued from previous page

Scrum Rules Unit Tests Documentation.pdf	Test cases documentation for each rule.
https://github.com/mirlindm/MasterThesis-Mirlind-2021/blob/main/Study%20Validation/Scrum%20Rules%20Unit%20Tests%20Documentation.pdf	

II. Additional Figures

Rule ID	Scrum Rule Definition as per Scrum Guide, Scrum Alliance, Scrum.org	Scrum Rule Interpretation	Theoretical Calculation of Rules
R1	"Sprints are fixed length events of one month or less." (Schwaber and Sutherland, 2020, p. 7, para. 3)	No more than five weeks should elapse for a single sprint.	Given the data fields from the dataset, we can check this Scrum rule using the *sprint* data field. This field contains information about each sprint, including the start time and end time. However, for some of the projects, this information is not present, meaning that for those projects we can't check this rule programmatically.

Figure 7. Scrum Rules Snippet (1)

Data Fields Used	Derived Variables	Scrum Phase Affected	Scrum Event/Artifact Affected	Scrum Role Affected	Verifiable	Not Verifiable
1. sprint	SPRINT_START, SPRINT_END, SPRINT_DIFFERENCE, MAX_LENGTH	The Game Phase	Sprint	Developers, Product Owner, Scrum Master	☑	☐

Figure 8. Scrum Rules Snippet (2)

R37 - There should be a minimum of one issue, representing a Sprint Backlog Item, per each Sprint

```
# THIS RULE CHECKS IF ANY OF THE SPRINTS WITHIN THE OPEN-SOURCE PROJECTS
# CONTAINS NO ISSUES, i.e. NO PRODUCT BACKLOG ITEMS/SPRINT BACKLOG ITEMS

print(f"\n The sprint with most completed issues is: {df.groupby('sprint').count().key.idxmax()}\n")

counts = df[['key', 'sprint']].groupby(['sprint']).describe()
display(counts)
print('\n\n')

pbis_per_sprint = df.groupby(['sprint'])['key'].apply(lambda grp: list(grp.value_counts().index)).to_dict()
print(pbis_per_sprint)



value = input("Check a number of PBIs that sprints might have: ")
total_pbis_per_sprint = [len(v) for v in pbis_per_sprint.values()]


if pd.to_numeric(value) == 0:
  if pd.to_numeric(value) in total_pbis_per_sprint :
    print(f"\nYes, this project contains sprints which have {value} PBIs!")
    print(colored.fg("red") + "\nRule 42 failed for this Open-Source Project!\n")
  else:
    print(f"\nNo, this project does not contain sprints with {value} PBIs!")
    print(colored.fg("green") + "\nRule 42 passed for this Open-Source Project!\n")
elif pd.to_numeric(value) != 0 and pd.to_numeric(value) in total_pbis_per_sprint:
  print(f"\nYes, this project contains sprints which have {value} PBIs!")
else:
  print(f"\nNo, this project does not contain sprints with '{value}' PBIs!")
```

Figure 9. Rule Implementation Snippet

R37 - There should be a minimum of one issue, representing a Sprint Backlog Item, per each Sprint.

```
%%file test_R37.py
print("Testing The Scrum Rules: R1")

# Import pandas library
import pandas as pd
import numpy as np


def test_R37_shouldPassWithMoreThan1PBIPerSprint():
  # initialize list of lists
  data_pass = [
          ['issue_1', '2020-03-01T11:10:08.305', '2020-03-02T11:10:08.305', 'sprint_1', 'XD'],
          ['issue_2', '2020-03-04T11:10:08.305', '2020-03-07T11:10:08.305', 'sprint_1', 'XD'],
          ['issue_3', '2020-03-10T11:10:08.305', '2020-03-12T11:10:08.305', 'sprint_1', 'XD'],
          ['issue_4', '2020-03-15T11:10:08.305', '2020-03-20T11:10:08.305', 'sprint_1', 'XD'],
          ['issue_5', '2020-03-20T11:10:08.305', '2020-03-21T11:10:08.305', 'sprint_1', 'XD'],
          ['issue_6', '2020-03-22T11:10:08.305', '2020-03-23T11:10:08.305', 'sprint_1', 'XD'],
          ['issue_7', '2020-03-24T11:10:08.305', '2020-03-25T11:10:08.305', 'sprint_2', 'XD'],
          ['issue_8', '2020-03-26T11:10:08.305', '2020-03-27T11:10:08.305', 'sprint_2', 'XD'],
          ['issue_9', '2020-03-28T11:10:08.305', '2020-03-29T11:10:08.305', 'sprint_2', 'XD'],
          ['issue_10', '2020-04-01T11:10:08.305', '2020-04-02T11:10:08.305', 'sprint_2', 'XD'],
          ['issue_11', '2020-04-03T11:10:08.305', '2020-04-04T11:10:08.305', 'sprint_2', 'XD'],
          ['issue_12', '2020-04-05T11:10:08.305', '2020-04-06T11:10:08.305', 'sprint_3', 'XD'],
          ['issue_13', '2020-04-07T11:10:08.305', '2020-04-08T11:10:08.305', 'sprint_3', 'XD'],
          ['issue_14', '2020-04-09T11:10:08.305', '2020-04-10T11:10:08.305', 'sprint_3', 'XD'],
          ['issue_15', '2020-04-11T11:10:08.305', '2020-04-12T11:10:08.305', 'sprint_3', 'XD'],
          ['issue_16', '2020-04-13T11:10:08.305', '2020-04-14T11:10:08.305', 'sprint_3', 'XD'],
          ['issue_17', '2020-04-15T11:10:08.305', '2020-04-16T11:10:08.305', 'sprint_3', 'XD']
          ]

  # Create the pandas DataFrame
  df = pd.DataFrame(data_pass, columns = ['key', 'created', 'resolutiondate', 'sprint', 'project'])
  pbis_per_sprint = df.groupby(['sprint'])['key'].apply(lambda grp: list(grp.value_counts().index)).to_dict()
  total_pbis_per_sprint = [len(v) for v in pbis_per_sprint.values()]
  result = (total_pbis_per_sprint > 0)
  assert len(total_pbis_per_sprint) != 0
  assert result == True
```

Figure 10. Unit Test Snippet

Scrum Rules Implenentation Results **BINARY OUTPUT**	XD 3152 issues 66 sprints	APSTUD 825 issues 34 sprints	TISTUD 2826 issues 59 sprints	MOBILE 3242 issues 23 sprints	MDL 63273 issues 269 sprints	DNN 1874 issues 103 sprints	MESOS 1365 issues 70 sprints	MULE 1278 issues 104 sprints	NEXUS 1037 issues 70 sprints	TIMOB 1853 issues 63 sprints	Final Sum / Total No of Projects %
R1: No more than five weeks should elapse for a single sprint.	N/A	N/A	N/A	0	0	N/A	N/A	0	N/A	N/A	**0**
R2: The number of the Scrum Team members per project should not be largely less or more than 10.	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	
R3: There should be only three different roles employed within a Scrum Team.	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	
R4: The duration of all sprints should follow similar pace.	N/A	N/A	N/A	0	0	N/A	N/A	0	N/A	N/A	**0**
R5: The next Sprint execution should begin only after the previous Sprint's resolution.	N/A	N/A	N/A	0	0	N/A	N/A	1	N/A	N/A	**0.33**
R6: There should be a project clarity identifier attached to each issue.	1	1	1	1	1	1	1	1	1	1	**1**
R7: No considerable amount of time should elapse between the finish of a sprint and the beginning of the new sprint.	N/A	N/A	N/A	0	0	N/A	N/A	0	N/A	N/A	**0**
R8: There should not be a considerable amount of time for a developer to volunteer and start a new issue after she/he has completed the previous one.	0	0	0	0	0	0	0	0	0	1	**0.1**

Figure 11. RQ2 & RQ3 Results Snippet

III. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Mirlind Murati**,
*(*author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Conformance Checking of Scrum Practices: A Study of 10 Open-Source Projects,
   *(*title of thesis)

   supervised by Ezequiel Scott, PhD and Fredrik P. Milani, PhD.
   *(*supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Mirlind Murati
04/08/2021