Universiteit Leiden
The Netherlands

# Master ICT in Business and the Public Sector

## Quantifying team performance by process mining Scrum board data: An architectural design and case-study from data extraction towards a machine learning application


| | |
|---|---|
| Name: | Aäron Kannangara |
| Student ID: | s1366106 |
| Date: | August 3, 2022 |
| 1st supervisor: | Dr. Christoph Stettina |
| 2nd supervisor: | Prof. dr. Joost Visser |


Master's Thesis


Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

**Abstract**

As software becomes increasingly complex, and its demand keeps growing, development teams are continually looking for ways to improve how software is created. Solutions to increased complexity and size can potentially be found in the large amount of data left by tooling used by development teams. Utilized tools store digital footprints of actual actions performed by team members throughout these processes.

Within this paper, an architectural design and case study using this design is presented to identify areas of Scrum improvement and measure transformations for an agile software development team. Using process mining from a control-flow perspective, the actual workflows of a .NET software development team working on large projects are analyzed and compared with the team's desired workflows. Event logs are generated by extracting data from Jira, the project management tool used by this team. The generated event logs are used during process discovery to create Petri net models of team workflows. Extending the event logs with data gained from the process discovery stage, data sets can be created with which to make predictions. The target values (issue work ratio, issue fitness, and issue next status) were predicted using a decision tree regressor, support vector regressor, and neural network. Unfortunately, the machine learning techniques were unable to accurately predict the target variables. Despite this, a proposed way to use accurately trained machine learning techniques to provide workflow recommendations is provided.

The contribution of this paper is threefold. First, an architectural design, based on the literature, for process mining Jira Scrum board data and providing recommendations is developed. Second, this architectural design is implemented in the form of a prototype process mining tool. Finally, by evaluating different process mining algorithms, a recommendation is made as to which process miner should be used and why. By comparing different control-flow miners, the data suggests that the heuristics miner is the most appropriate to use when analysing agile software development team data as well as that there is a missing fifth model quality metric: usability.

# Contents

# 1 Introduction

We live in a world where there is an increasing dependency on IT systems. Since the latter part of the 1990s, digitalization was a key issue for governments and businesses (Anttiroiko, Valkama, & Bailey, 2014). Everything from banking to sending mail or packages use IT systems to help facilitate and coordinate the processes. These system digitization trends have continued to evolve over the last years, becoming increasingly complex, especially when taking into account elements like the internet of things (IoT), web services, cloud- and edge computing (Gill et al., 2019).

As technological developments continue, so does the need for firms to actively integrate digital technology into products, services, and operations to achieve - and then sustain - a competitive advantage in their market (Koch & Windsperger, 2017). This requires felicitous digital service design. The application of artificial intelligence (AI) and big data analytics can then be used to solve business problems and this application "is growing in leaps and bounds" (Chui, Manyika, & Miremadi, 2018).

While the different types of technology are fast developing, organizations are often very slow in nature and rigid in thinking. Despite their inherent inflexibility, organizations do need to take advantage of opportunities provided by technology to facilitate continual competitiveness in the market. As such firms are increasingly implementing environmental technologies; which are "production equipment, methods and procedures, product designs, and product delivery mechanisms that conserve energy and natural resources, minimizing the environmental load of human activities" (Shrivastava, 1995). Conservation of energy and resources, including human resources, along with process improvement can help reduce cost. Energy and human resource application reduction are some of the types of problems AI and big data analytics can help with.

Despite the intrinsic interest of organizations in cost reduction, it is widely known that most software development projects suffer from time- and cost-overrun. This naturally conflicts with the desire to reduce human or other resource expenditure, and with the strategic and market need to satisfy customers.

An attempt to approach, and solve, the problem of time-overrun is found in Development Operations (DevOps) which aim to automate and structure certain processes to reduce time on repetitive tasks. This is often done by way of CI/CD (Continuous Integration/Continuous Deployment), where a central code repository is used for version control and deployment (to production) is automated. Part of DevOps is also to define and improve team standards and workflows.

AIs have proven effective in industries like manufacturing (Monostori, 2003) or around supply chain management (Souza, 2014) in reducing cost as well as improving processes. The question is then how similar principles of process improvement and cost reduction can be applied to digital service design to help organizations take advantage of the incrementally growing technology and technological possibilities?

The Netherlands has a highly digitalized economy, being, as of writing, the fifth largest data center nation in the world (Cloudscene, 2018). To lead in the next technological wave, Dutch companies need to be better in the digital service domain than their competing nation economies. IT development and service delivery development provide enhanced strategic capabilities for many companies

and governments, enabling further sustainable growth.

Electricity grid companies might need to double their energy output or web development companies might need to increase their client turnover. Digital service design can help reach these goals by making work easier and less resource-consuming while still maintaining or even increasing output. To apply digital service design in such a way that it helps companies reach their goals, digital service design itself needs to be improved through cost and time reduction by way of process improvement. Setting up a digital service delivery team is often difficult: forming and training a professional team is expensive while picking up on and applying best practices takes time and money. Scaling these best practices becomes difficult due to constant domain developments and new best practices being defined.

While process improvement through AI-driven process mining has been applied in areas like manufacturing (Lee, Choy, Ho, & Lam, 2016), its application is viewed as less suitable for project management and service design primarily due to the "inherent uniqueness of projects by definition" (Auth, JokischPavel, & Dürk, 2019). Despite this issue, many different design and development life-cycle methods have been proposed like agile and waterfall. This inherent uniqueness makes it difficult to define a single straightforward process that all digital service delivery teams should follow to deliver any relevant products. This also often leads to digital service design teams following radically different processes to deliver the same type of product.

The applications of (AI-driven) process mining have been implemented to tackle many different business problems including manufacturing optimization (Lee et al., 2016), healthcare (Rojas, Munoz-Gama, Sepúlveda, & Capurro, 2016), supply chain management (Lau, Ho, Zhao, & Chung, 2009), construction (Van der Aalst et al., 2007), and financial services (De Weerdt, Schupp, Vanderloock, & Baesens, 2013). In these industries, process mining aims to construct models reflecting observed behavior and processes while also allowing for bottleneck identification and process optimization opportunities.

Business analytics applications for project management have been categorized into descriptive, predictive, and prescriptive analytics (Souza, 2014), the latter two of which are often heavily AI dependent. Descriptive analytics aims to present and describe the project itself. Predictive analytics attempts to forecast project outcomes based on resource allocation and the current situation. This can be useful in predicting the required time and cost and is heavily dependent on previous data. Finally, prescriptive analytics is generally regarded as a less well-matured extension of descriptive and predictive analytics (Hagerty, 2017) and has gained increasing research interest over the last few years (Lepenioti, Bousdekis, Apostolou, & Mentzas, 2020). Answering what to do and why to do it is still a problem and is considered "the next step towards increasing data analytics maturity and leading to optimized decision making, ahead of time, for business performance improvement" (Lepenioti et al., 2020).

Despite the difficulties of defining set processes to inherently unique projects, like those taken on by digital service design teams, this paper aims to provide an example architectural overview and implementation, building upon industry standards, in applying process mining combined with machine learning techniques to provide descriptive, predictive, and prescriptive analyses of the processes these teams go through.

## 1.1 Research Objective

Digital service design teams often focus on the design, development, and delivery of digital services in the form of computer programs or (web-)applications, like APIs that provide a central connection point between different applications. Since the development of the first computer program, arguably by Ada Lovelace in 1834, many different software development life-cycle methodologies have been proposed. The different suggested methodology styles has exploded since the 1970s, leading to common methodologies like Waterfall, Agile, and more recently SAFe and DevOps. The most commonly applied software life-cycle methodology used nowadays is agile, which in itself often encompasses other methodologies and elements like Scrum and DevOps.

As these methodologies have become popular and widely used by the growing number of software development teams so have the number and capabilities of digital tools that facilitate these processes while also logging the steps taken by these development teams. One of these widely used tools is the online project management software tool Jira.

Process mining is a promising field of study in analysing event data and has been growing rapidly in popularity over the last two decades(Van der Aalst, 2012b; Homayounfar, 2012). Despite its repute, process mining can seem like a daunting task, especially when applied in process improvement for software development. *The objective of this paper will be to provide an architectural design for process mining of agile software development teams Scrum board workflow data based on relevant literature and best practices combined with a case-study applying this proposed framework built in Python.*

A case study will be used to investigate the Jira Scrum board logs of an agile software development team. During this investigation, the logs will be used to provide process mining driven descriptive and predictive analysis and attempt to provide prescriptive advice for this team. This analysis and advice will be developed using (control-flow) process mining of the Scrum board logs combined with the industry-standard machine learning techniques. The application will be built completely in Python.

## 1.2 Research Questions

Based on the above-stated research objective, this research paper will aim to answer the following question through a case study of an agile software development team's project management software (Jira) event log data:

**How can process mining be used to suggest Scrum improvements for an agile software development team, and what kind of architectural design is required to investigate and provide these recommendations?**
This broad thesis question can be further divided into, and extended with, the following supporting questions:

1. What does process mining entail, what kinds of process mining exist, and what are the industry standards?
2. How do process mining techniques and standards compare?
3. How is process mining applied in (agile) software development?

4. What machine learning techniques are industry standards in analyzing process mining data and how do they fare during the case study?

## 1.3 Research Scope

In this exploratory research paper, a single agile software development team has been identified and its Jira Scrum board event log data is extracted and analyzed. This research will therefore be structured as a single-case study over the course of about a year. Despite the definitive data extraction being done towards the end of the research period, writing the paper, collecting data, and building a Python application to perform the research will take around a year, after which the researcher will be expected to graduate.

With this research, the aim is to analyze data of an actual software development team working in the industry. For this reason, several restrictions will be placed on the type of team to analyze. The requirements of the type of team to analyze are as follows:

- Agile software development team (what this entails is discussed in Section 2).
- Uses Jira as their project management tool.
- At least data of 10 projects to analyze.
- Preferably working on large-scale projects (multiple components developed across multiple teams) - working with multiple teams on a single large project.

Most process mining research is done using frameworks and tools like ProM or Celonis, which in themselves do not have machine learning capabilities built-in. Machine learning research is often done in Python. For this reason, a Python application is developed and published on Github to allow the combination of process mining (using PM4PY) and machine learning (using Scikit Learn) in one place. This application, with which to extract, explore and analyze data, is part of this research paper and an architectural design for this application is also presented to help guide any further research in this area that wishes to combine machine learning and process mining in a single application.

## 1.4 Project overview

This paper will consist of six sections:

- Section 1: The introduction, which you just finished reading, contains the research objective, question, and scope.
- Section 2: The literature review contains background information to help guide understanding and analysis of the data while also aiming to answer some of the supporting research questions.
- Section 3: The method, where the methodology of the research will be discussed in detail.
- Section 4: Architecture of the Python application that will be built to combine Jira data collection, process mining, and machine learning into a single application.
- Section 5: The experiments and results of the experiments will be presented.

- Section 6: The discussion of the results where the results will be interpreted and an understanding of the meaning of the results will be formed.
- Section 7: The conclusion, where the main findings will be presented relative to the objects of this research paper.

# 2 Literature Review

For the literature review, this Section will be split into several subsections in an attempt to approach and answer several of the supporting research questions. For this reason, this Section will define and describe different software development methodologies, what process mining entails, and how it is applied in agile software development. Finally, a summary and gap analysis to bridge the presented knowledge, answering several of the supporting research questions.

## 2.1 Agile and other software development methodologies

As early as the 1970s software development life-cycles have been defined as a way of alleviating the issues that arise from software projects with growing complexity. Royce defined what would become the classic waterfall model in 1970(Royce, 1987) which was later redefined and coined as such in 1976 by Boehm(Boehm, 1976).

Every software development model acts as a set of rules or guidelines while moving through the general steps of building a system(Despa, 2014). As software projects become more complex they often lead to general problems. These problems include cost overrun, time overrun, the software being unreliable, and the software failing to meet the users' needs or expectations(Davis, Bersoff, & Comer, 1988).

These types of problems often arise from incorrect work and resource alignment by the project director(Bassil, 2012). Phases are delayed because of insufficient resource allocation or phases become idle by way of excess resource allocation, causing bottlenecks between the phases and their sub-elements, eventually leading to operational project delivery failure through time and cost overrun(Bassil, 2012).

By defining a structure for the software development process, these models aim to tackle some of these problems that arise in complex projects. The waterfall method, for example, approaches these problems by encouraging requirements definition and system design *before* building; planning of component interactions; tracking development progress to identify overrun early; documentation to assist in testing and maintenance; reducing maintenance costs; and stimulating structured and manageable system development(Davis et al., 1988).

The approaches suggested by different development models aim to help the project manager structure the project in such a way as to avoid time and cost overrun. To this end, the models aim to maximize development process utilization by keeping all team members and resources busy while keeping the pace of project elements and decreasing idle time(Bassil, 2012).

As many as 32 different software development models have been defined(Despa, 2014). Each of these models have their own respective advantages and disadvantages. While different in nature, seven broad stages have been identified and agreed upon to be part of every software development process. Each of these stages can be defined by having its own deliverables, specific time frame, and can be assigned a weight of importance as compared to the other stages in the overall project. Every project contains a research, planning, design, development, testing, setup, and maintenance stages(Despa, 2014).

Despite many different models being defined, these models can be divided up into two distinct types. Firstly, there is the more traditional heavy weight model type. These models are "derived from the waterfall model and emphasizes detailed planning, exhaustive specifications and detailed application design"(Despa, 2014). This coincides with projects where detailed planning is permitted by the required software complexity and requirements are unlikely to change. Secondly, there are the light weight model types which are "derived from the agile model and promote working software, individuals and interactions, acceptance of changing requirements and user feedback"(Despa, 2014). Contrary to heavy weight models, light weight models work well for projects where software complexity is difficult to estimate and requirements can or are likely to change.

The waterfall methodology, described by Royce, is the first project development model to generally be accepted as dedicated to software development(Bassil, 2012). Being the precursor for many other models, it is important to understand the workings of the waterfall method. While several different namings for the stages exist, the waterfall model is structured in sequential stages, each of which have their own deliverable and which reflect the stages present in every software project: Research, planning, design, development, testing, feedback, setup, maintenance. A feedback stage has been added between testing and setup when compared to the original workflow of the waterfall methodology. These stages can be viewed in Figure 1.

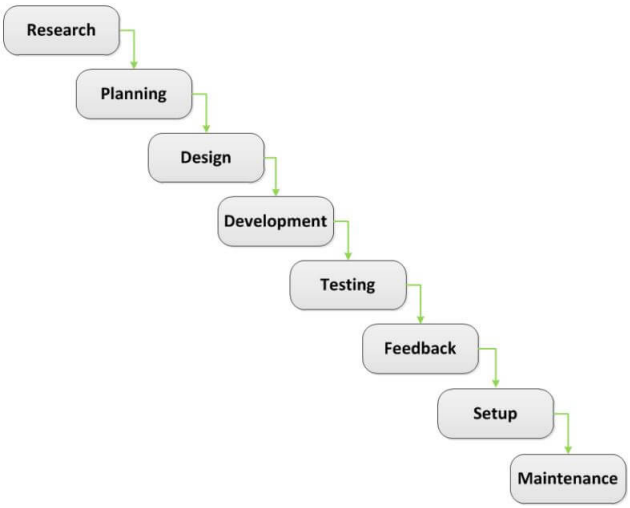



Figure 1: Waterfall development stages(Despa, 2014)

While the stage name generally describes the workings of that stage, the research stage requires extra attention due to its importance in defining the project scope and success, especially important the waterfall model. Exhaustive planning only works if the right amount of knowledge of the project and its dependencies is present at the start. During the research stage, functional and non-functional requirements are set, helping to set the scope of the project. Functional requirements are the "use cases which describe the users' interactions with the software"(Bassil, 2012), while non-functional requirements consist of "various criteria, constraints, limitations, and requirements imposed on the design and operation of the software rather than on particular behaviors"(Bassil, 2012).

Feedback from the project owner is only given after development and testing, which makes the planning and (non-)functional requirements definition extremely important. The nature of this development model makes applicable during small-scale projects with clear requirements and where detailed planning is easily drafted(Despa, 2014).

Unfortunately, not all software development projects allow for clear requirements definition. In fact, it is very common for functional or non-functional requirements to change during software

development. Some would even go so far as to say "software development projects are notorious for frequently changing initial specifications"(Despa, 2014).

Requirements can change for several reasons: new business opportunities are identified by the project owner; there can be a lack of consensus around the expected outcome, spurred by the technical nature of the project; the original specifications were misunderstood or poorly illustrated; technical limitations or lack of experience on the part of the development team; changes in the context of software usage; or simply the launch of new technologies in market place (Despa, 2014).

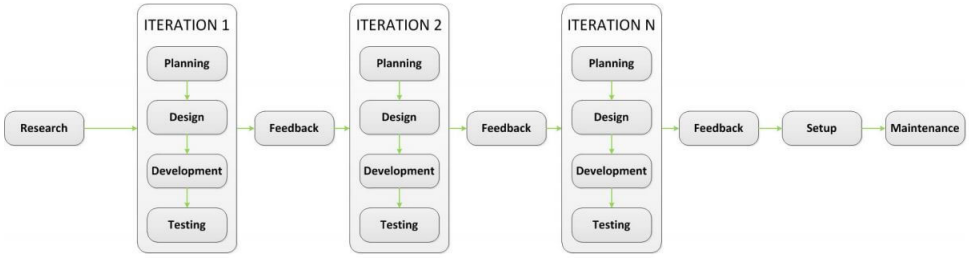



Figure 2: Iterative and incremental development model(Despa, 2014)

As a way to keep up with changing (non-)functional requirements and project specifications, iterative and incremental development models were suggested. These light weight models emphasized building the project one step at a time, in an iterative manner, enhancing and expanding the system as development continued. In this way modules of the system were not disregarded or redefined but extended(Davis et al., 1988). Each iteration would consist of its own planning, designing, development, and testing phases(Davis et al., 1988). After each iteration a fully functional system can be presented for feedback. These types of models gave birth to agile development models as agile methods are regarded as a branch of iterative methods(Larman, 2004).

| | **AGILE** | **TRADITIONAL** |
|---|---|---|
| **User requirement** | Iterative acquisition | Detailed user requirements are well-defined before coding/implementation |
| **Rework cost** | low | high |
| **Development direction** | Readily changeable | Fixed |
| **Testing** | On every iteration | After coding phase completed |
| **Customer involvement** | high | low |
| **Extra quality required for developers** | Interpersonal skills & basic business knowledge | Nothing in particular |
| **Suitable Project scale** | low to medium-scaled | Large-scaled |

Figure 3: Comparison between Agile and Traditional (Waterfall) methods(Leau et al., 2012)

Agile software development is focused around a manifesto consisting of four major pillars(Lau et

al., 2009). First, there is early, active and continual customer involvement to ensure alignment between the customers evolving wishes and the development team. Secondly, development should take place in an iterative manner, involving the customer actively after each iteration. Thirdly, the development team should self-organize to allow for the most creative development process. Finally, an agile method requires active adaption to change, welcoming change over renouncing it.

The agile manifesto suggests twelve agile principles that are supposed to guide the development team. These principles are as follows(Williams, 2010):

1. "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software."
2. "Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage."
3. "Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter time scale."
4. "Business people and developers must work together daily through the project."
5. "Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done."
6. "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation."
7. "Working software is the primary measure of progress."
8. "Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely."
9. "Continuous attention to technical excellence and good design enhances agility."
10. "Simplicity - the art of maximizing the amount of work not done - is essential."
11. "The best architectures, requirements, and designs emerge from self-organizing teams."
12. "At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly."

The agile principles would later be extended and tuned by Mary and Tom Poppendieck in 2003 in the form of the lean principles(Poppendieck & Poppendieck, 2003). Lean development became popular in the 1990s and referred to Toyota and Honda automobile manufacturing principles(Womack, Jones, & Roos, 2007). The Poppendiecks definition of the lean principles are listed below and a comparison of the Agile principles with the lean principles can be found in Appendix A.

1. "Eliminate waste"
2. "Build quality in".
3. "Create knowledge"
4. "Defer commitment"
5. "Deliver fast"
6. "Respect people"
7. "Optimize the whole"

As many as 32 agile practices can be applied by development teams and development teams define and integrate their own subset of these practices(Williams, 2010). Popular groupings of these practices gave rise to different agile software development methodologies. The most used agile methodology is by far the Scrum methodology(Moniruzzaman & Hossain, 2013), followed not-so-closely by a Extreme Programming(XP)-Scrum hybrid, see Figure 4. All of these agile methodologies have their own defined subset or extension of the above mentioned 32 agile practices.

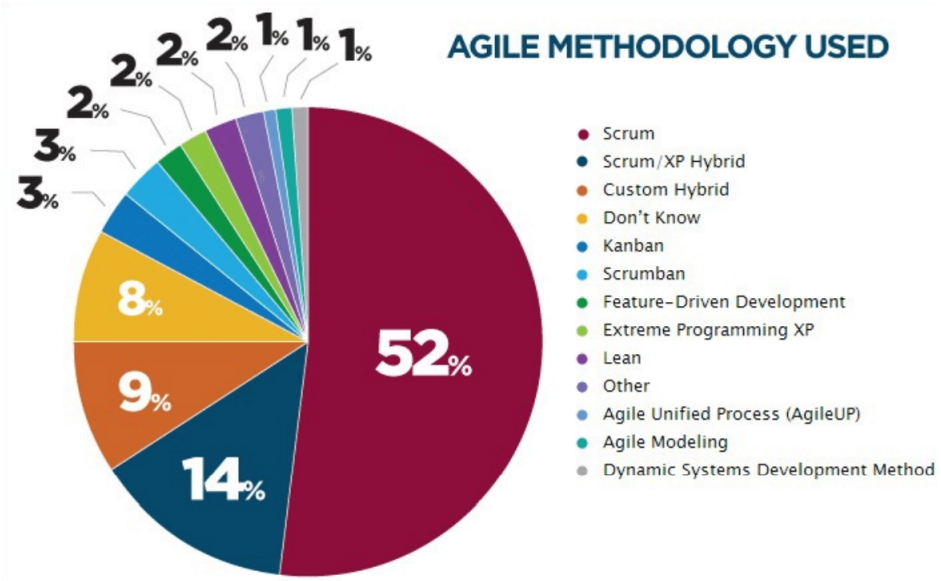
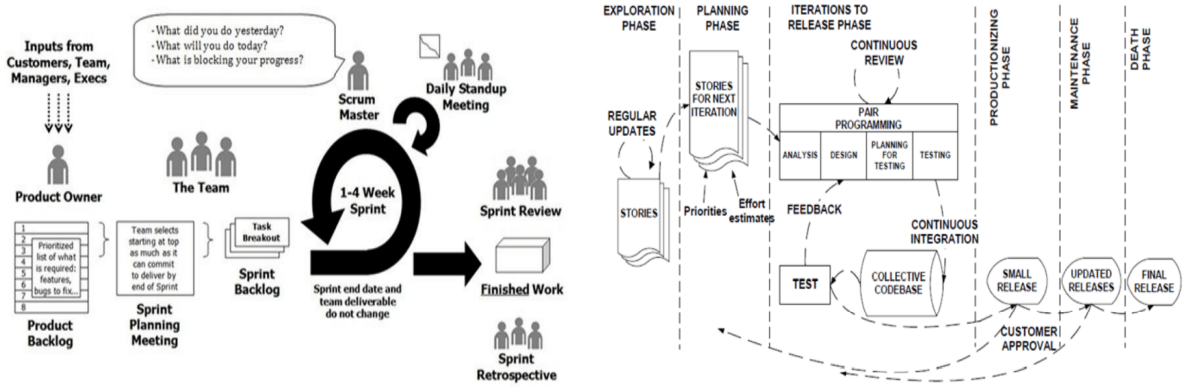



Figure 4: Comparison usage of agile software methodologies (Moniruzzaman & Hossain, 2013)

The scrum methodology was initially introduced in 1995 by Jeff Sutherland and Ken Schwaber. Six years later, in 2001, it was codified in "Agile Software Development with Scrum", a book by Schwaber and Mike Beedle. Kent Beck defined Extreme Programming (XP) in 1996, publishing a book defining the methodology in 1999. Both of these light weight and iterative software development methodologies are aimed at tackling the seemingly unavoidable problems in the development process like high project failure rates, cancellations, and project delays(Anwer, Aftab, Shah, & Waheed, 2017). The general life cycle of both of these methodologies are presented in Figure 5 and follow the above mentioned general development stages in an iterative manner.




(a) Scrum lifecylce

(b) XP lifecycle

Figure 5: XP and Scrum lifecycles (Anwer et al., 2017).

While there are many agile practices, and many alterations on these practices, some standout due to their importance in popular methodologies. Both Scrum and XP have a product or sprint backlog, for example, which entails "an evolving, prioritized queue of business and technical requirements that needs to be developed into a system during the release"(Williams, 2010) where each item is placed in a category, is given a priority value, is assigned a resource usage and/or time estimate and needs to be unique for that project.

Items in the backlog can be defined as Story Cards or Features. This aims to clearly define and structure the task. As a story, for example, the task is structured as a $< \ usertype \ >$ wanting to $< \ performaction \ >$ to achieve a $< \ result \ >$; for a feature it could be $< \ action \ >< \ leads \ to \ >$ $< \ object \ >$. In this way non-ambiguous requirements are set for the task so that the developer can immediately see what needs to be done; simplicity is key. If this feature is too large to develop by the team within a set maximum amount of time or within a single iteration, the task should be made into an *epic* and split into subtasks. Tasks are subsequently assigned resource or time estimations during the (iterative) planning phases in both Scrum and XP. In Scrum, this is done with activities like Planning Poker in which the team uses collective knowledge to make estimates.

The tasks are then tackled during Sprints which are a defined time slots, representing a single development iteration, of one to four weeks in which tasks are completed such that after each iteration a fully functional (although maybe not complete) system can be presented. During a sprint iteration, the backlog for the next iteration should be filled. Both XP and Scrum introduce Standups (called scrum meetings in Scrum) in which the team gets together daily for at most 15 minutes to discuss what they have done, what they will do, and what problems they are running into. After each sprint, a retrospective meeting can be performed to discuss what went well and could be improved from the just completed sprint.

XP goes on to emphasize the importance of Pair Programming, in which two developers work together on a single task from one machine(Anwer et al., 2017). This aims to improve code quality as well as increase task turnover.

Both Scrum and XP have their own defined roles within the development process. XP defines a

Manager, Coach, Tracker, Programmer, Tester, and Customer. Scrum defines a Product Owner, Scrum Master, and Developer. For both XP and Scrum a team member can sometimes be assigned multiple roles, for example in Scrum a Developer can also be the Scrum Master and in XP a Programmer can also be a Tester.

Both XP and Scrum can apply Done Criteria to tasks as well as demand Continuous Integration, Automated Testing, and Coding Style Guidelines. A comparison of XP and Scrum can be found in Appendix B. The application of agile principles in XP and Scrum allow for more flexible and adaptive development emphasising simplicity and agility; exactly what the agile principles and practices aim to enforce to increase customer satisfaction, enable the ability to process changing requirements, and increase the project success rate.

## 2.2 Process mining

Process mining is a relatively young research field which has been growing over the last two decades(Van der Aalst, 2012b; Homayounfar, 2012). The main objective of process mining is to "discover, analyze, and improve business processes based on event data" (Van der Aalst, 2012a). In 2012, van der Aalst and colleges released a *Process Mining Manifesto* (Van der Aalst et al., 2012) defining terms and general process mining principles. This manifesto aims to guide stakeholders, like developers, business managers, and end-users, in improving "the (re)design, control, and support of operational business processes"(Van der Aalst et al., 2012).

Different activities performed by people or software often leave some form of footprint or trace in the form of an event log(Van der Aalst, 2012a) which are required for all forms of process mining. These event logs record sequential *events*, where each event reflects a well-defined step (*activity*) in the process and is associated with a specific *case* (process instance)(Van der Aalst et al., 2012). The manifesto also encourages the possibility to extend these event logs with defined *resource(s)* (person executing the task, for example), *timestamp* of when the activity was executed, and other *data elements* further defining the event.

As mentioned in the *Process Mining Manifesto*, by using event logs, processes can be identified, analyzed, and possibly improved. Van der Aalst et al. defined three types of process mining with regard to input and output: *discovery*, *conformance checking*, and *enhancement*(Van der Aalst et al., 2012). Dos Santos Garcia et al. extend these three categories with a fourth: *supporting areas*(dos Santos Garcia et al., 2019). All of these categories will be described in the following sub-sections.

### 2.2.1 Process Discovery

During process mining discovery, an event log is used to extract knowledge, identify sequential activities and develop a control-flow model without prior knowledge or information about the process(es). The actual processes teams use are revealed using only the behavior stored in the event logs(Van der Aalst, 2012b). In this way, process discovery can help managers identify the executed cases, frequency paths between activities, and how path frequency is distributed over all the paths(De Medeiros & van der Aalst, 2008).

Process discovery aims to produce a control-flow-model that should be sufficiently comprehensive and accurate with regard to the listed cases in the event log while avoiding unneeded complexity (De Weerdt, De Backer, Vanthienen, & Baesens, 2012). To this end, a balance between recall accuracy, precision, and generalization is intended(dos Santos Garcia et al., 2019) where the F-score methodology is used to assess model accuracy(De Weerdt et al., 2012). The suggested models are developed using process discovery algorithms or 'miners'.

Hundreds of different types of miners have been proposed over the years(dos Santos Garcia et al., 2019). As early as 1972 Biermann and Feldman's suggested the KTAIL algorithm (Biermann & Feldman, 1972). Other example algorithms include the in 1995 by Cook and Wolf suggested Recurrent Neural Network, and a, statistically based, Markov model for process discovery on finite state machines(dos Santos Garcia et al., 2019). Since then many new algorithms have been suggested including: Agrawal (Agrawal, Gunopulos, & Leymann, 1998), Alpha miner (Van der Aalst, Weijters, & Maruster, 2002), Heuristic miner (A. J. Weijters & Van der Aalst, 2003), social miner (Van der Aalst & Song, 2004), genetic miner (Van der Aalst, De Medeiros, & Weijters, 2005), Integer linear programming (ILP) miner (Jansen-Vullers, van der Aalst, & Rosemann, 2006), Fuzzy miner (Günther & Van Der Aalst, 2007), Declarative miner (Lamma, Mello, Montali, Riguzzi, & Storari, 2007), Region-based miner (Bergenthum, Desel, Lorenz, & Mauser, 2007), Inductive miner (Leemans, Fahland, & van der Aalst, 2013), Multi-paradigm miner (De Smedt, De Weerdt, & Vanthienen, 2014), and the Fodina miner (Van den Broucke & De Weerdt, 2017).

Each of these miners have their own respective benefits and disadvantages. By far the most popular miner is the heuristic miner, and its many variations, see Figure 6. This popularity could be because of the algorithms robustness and capability to "deal with noise and exceptions in unstructured processes"(dos Santos Garcia et al., 2019).

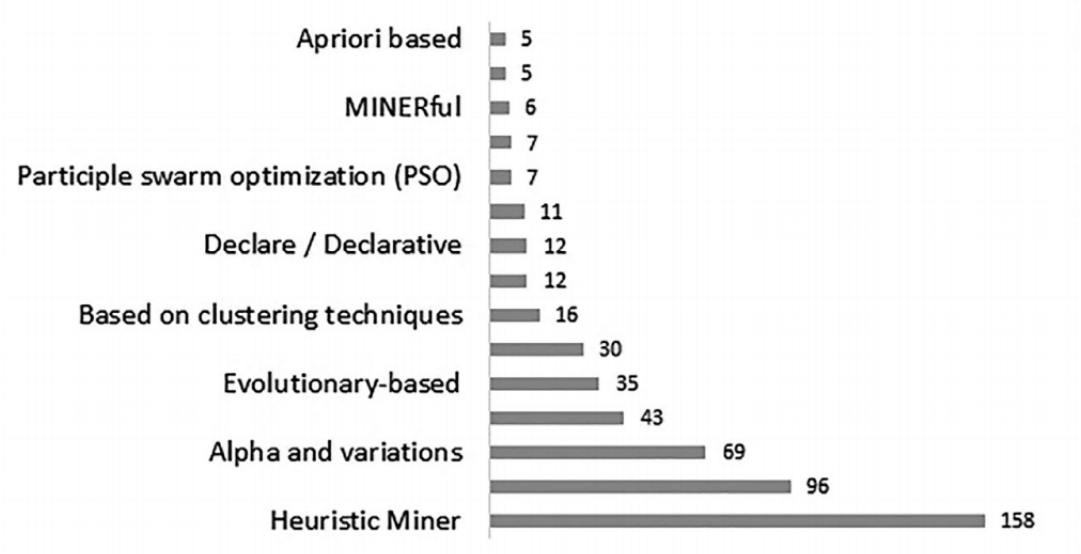



Figure 6: Distribution of process mining algorithms by use (dos Santos Garcia et al., 2019)

The type of miner to use is often based on the process mining perspective to be analyzed. The process mining manifesto identifies four general perspectives: *control-flow*, *organizational*, *case* and

*time* perspectives(Van der Aalst et al., 2012). While several alternative or extending perspectives have been provided, Rubin et al. neatly defines four over-arching perspectives(Rubin et al., 2007). First, the most applied perspective is the *control-flow perspective* where the process is defined by sequential, and sometimes looping, activities in a visual process model, like a Petrinet or a BPMN model. Secondly, the *resource perspective* (organizational perspective) analyzes the actors involved in the process and their relationships. This more closely resembles social network analysis(Gupta, Sureka, & Padmanabhuni, 2014) and can offer c-level support in employee allocation decisions(dos Santos Garcia et al., 2019). Next, there is the *performance perspective* from which the *timestamp* attribute of an *event* is used to identify performance frailty and it location in the process. Finally, the *information perspective* which aims to take a high level view of the activities, clustering similar activities to expose potentially hidden dependencies.

Several different miners have been applied in process discovery in (agile) software development from a control-flow perspective. Selecting a miner can be difficult, as different miners can offer similar but not equivalent results. To be able to evaluate and compare the different miners, Rozinat et al. proposed a framework consisting of four quality dimensions(Rozinat, de Medeiros, Günther, Weijters, & van der Aalst, 2007). The first metric is that of *fitness*, indicating how well the event data is captured by the proposed model. Second, *precision* measures how well the event log activity sequences fit into the model without the model over-generalizing (under-fitting). The third metric, *generalization*, does exactly the opposite, by assessing how well the model avoids over-fitting. The final metric, *simplicity*, is that of the structure of the proposed model, measuring how easy the model is to read. This final metric is heavily dependent on the vocabulary and formalism of the modeling language. These metrics become especially important when discussing the *conformance checking* type of process mining, see Section 2.2.2.

For this paper, we will only be describing the most popular control-flow miners. These include the heuristics, Alpha (and variations) and evolutionary-based miners, see Figure 6. We will also include the Inductive Miner due to its polynomial complexity and enforcement of soundness and fitness (more on this further below).

Often regarded as the most basic algorithm for process mining(Shani, Sarno, Sungkono, & Wahyuni, 2019), the Alpha miner proposed by van der Aalst(Van der Aalst et al., 2002) in 2002 aimed to discover workflow models by designing Petri nets during computation. Unfortunately, the initial proposal was unable to properly process incomplete event logs, identify short loops, map non-local dependencies, handle non-free choice constructs, and was simply non-resilient to noise(dos Santos Garcia et al., 2019). Many process miners aimed to extend the Alpha miner, the most important of which to mention is its namesake, the Alpha$^+$ algorithm introduced in 2006 by Wen et al.(Wen, Wang, & Sun, 2006). Alpha$^+$ solved some of the issues around its predecessor including the ability to process short loops, but was unable to properly process noise or calculate activity frequency(Shani et al., 2019).

The Heuristics algorithm is another control-flow discovery miner that was proposed by van der Aalst together with Weijters(A. J. Weijters & Van der Aalst, 2003) a year after the Alpha miner was presented. This algorithm aimed to further extend the Alpha algorithm and has certain advantages over the Alpha$^+$ algorithm(Shani et al., 2019). The heuristics miner is able to identify noise in the process(Zayed & Farid, 2016), analyze path frequency, use dependency frequency to develop a heuristic net starting from the event logs instead of the process model design (as done with

the Alpha miner). Dependency thresholds are used to help differentiate noise from low frequency pattern(A. Weijters, van Der Aalst, & De Medeiros, 2006). Three threshold values are defined: dependency threshold (direct sequential relationship), and threshold (looping to self sequentially), and finally two loop threshold ([indirect] sequential relationship)(A. Weijters & Ribeiro, 2011). The higher the threshold, the more often the path as to occur to be included in the model. While the model cannot guarantee total soundness, as less frequently visited paths are not included in the model (being limited by the threshold values), it does ensure good fitness and the ability to compare actual event processes with a handmade model(dos Santos Garcia et al., 2019). The benefits and robustness of this algorithm have allowed it to become the most used and extended process mining algorithm.

The in 2007 by Günther et al. proposed Fuzzy miner(Günther & Van Der Aalst, 2007) which further built upon the Heuristics miner. This new miner was designed to identify unstructured logged processes(Zayed & Farid, 2016) and visualize activity relationships. This method "combines complex topology concepts from cartography, such as aggregation, abstraction, emphasis, and customization"(dos Santos Garcia et al., 2019). Unfortunately, soundness and fitness are not ensured with the Fuzzy miner.

Two years before the Fuzzy Miner was presented by Günther et al., van der Aalst, Alves, de Medeiros, and Weijters proposed using Genetic Algorithms - a subdomain of evolutionary algorithms - for process discovery(Van der Aalst et al., 2005). Each individual (case) in the population was represented in the form of a two dimensional *casual matrix* which could be transposed into a petri-net. Using a genetic algorithm, the casual matrices of the individuals in the population were mutated, and a fitness function was used to simulate Darwinian survival of the fittest. This allowed for the identification of duplicate activities as well as silent activities(Van der Aalst et al., 2005). Silent activities are those that do not appear in the event log but still take place.

The final discovery algorithm to highlight is the Inductive miner, also called the B' algorithm (Leemans et al., 2013). This miner aims to identify both sound and fit block-structured models in polynomial-time. "It also incorporates several criteria such as: frequency analysis, clustering, detection of deviations and frauds, analysis of times and bottlenecks, general vision, and process understanding and evaluation of values by outcome"(dos Santos Garcia et al., 2019) into a single mining solution.

Despite the Inductive miner, given its polynomial complexity and enforcement of soundness and fitness, showing much promise over its alternatives it has not (yet) become the most prevalently used miner, as reflected in 6. The Heuristics miner and its descendant, the Fuzzy miner, are the most applied miners. Both of these miners have a resilience to noise in unstructured processes which might explain their popularity(dos Santos Garcia et al., 2019).

### 2.2.2 Conformance Checking

Having discussed the *discovery* type of process mining, the *conformance checking* side of process mining should be highlighted next. With conformance checking the quality of a process model is measured, applications of these quality measurements are defined and can then be used for active process monitoring. Model quality is measured along the four quality dimensions proposed by Rozi-

nat et al.(Rozinat et al., 2007): fitness, precision, generalization, and simplicity. These metrics have already been described above in Section 2.2.1.

While often used interchangeably, conformance and compliance checking differ slightly. Conformance checking requires analysis of completed event logs after the project(s) is(/are) finished, while compliance checking is a form of monitoring using uncompleted event logs *during* (project) run-time(Ly, Maggi, Montali, Rinderle-Ma, & van der Aalst, 2015). Despite having a subtle difference between them, these forms of checking are heavily influenced by the level of completeness and structure of the event logs being analyzed.

The conformance checking type of process mining is where c-level interest is sparked as this type focuses on conformance with established norms and identifying where rules or expectations are not followed. De Medeiros and van der Aalst approached the c-level questions that can be tackled with process mining from both the discovery as well as the conformance checking type perspectives(De Medeiros & van der Aalst, 2008). Questions like: "What is the communication structure and dependencies among people? Who are the important people in the communication flow? Are the process rules [and] constraints indeed being obeyed? What are the business rules in the process model? How compliant are the cases (i.e. process instances) with the deployed process models?"(De Medeiros & van der Aalst, 2008). The complete list of approachable question suggested by de Medeiros and van der Aalst are listed in Figures 7a and 7b.

| Question |
|---|
| How are the cases actually being executed? |
| What is the most frequent path for every process model?<br>How is the distribution of all cases over the different paths through the process? |
| How many people are involved in a case?<br>What is the communication structure and dependencies among people?<br>How many transfers happen from one role to another role?<br>Who are the important people in the communication flow?<br>Who subcontract work to whom?<br>Who work on the same tasks? |
| Are the process rules/constraints indeed being obeyed? |

(a) Questions to answer with process discovery

| Question |
|---|
| How compliant are the cases (i.e. process instances) with the deployed process models? Where are the problems? How frequent is the (non-) compliance? |
| What are the routing probabilities for each slipt/join task?<br>What is the average/minimum/maximum throughput time of cases?<br>Which paths take too much time on average? How many cases follow these routings? What are the critical sub-paths for these routes?<br>What is the average service time for each task?<br>How much time was spent between any two tasks in the process model? |
| What are the business rules in the process model? |

(b) Questions to answer with process conformance checking

Figure 7: Questions to ask and answer with process discovery and conformance checking as proposed by de Medeiros and van der Aalst(De Medeiros & van der Aalst, 2008)

De Medeiros and van der Aalst went on to describe how event logs and models can be used to represent relationships, defining different ontologies in the processes being analyzed. Ontologies can be summed up as "a set of concepts used by people to refer to things in the world and the relationships among these concepts"(De Medeiros & van der Aalst, 2008). Simply put, the *what* of the processes. Examples of these ontologies can be Task ontology, Role ontology, and Performer ontology, which are also presented in Figure 8. These ontologies are very similar to the different perspectives of process mining defined in the previous sub-section - the *control-flow*-, *organizational*-, *case*- and *time* perspectives.

Ten general functions have been identified and placed in a framework for the exercise of analyzing process conformance and exploring the relevant ontology. These are the *Compliance Monitoring Functionalities* (CMFs) which can be used to evaluate different compliance checking approaches. These ten functionalities have been defined by Ly et al. along the following constraints: time constraints; data constraints; resource constraints; supporting atomic (spanning one time interval) and non-atomic activities; supporting activity lifecycle; multiple-instance constraints; reactively detecting and managing compliance violations; pro-actively detecting and managing compliance violations; explaining the cause of the violation; and finally, quantifying the degree of compliance(Ly, Maggi, Montali, Rinderle-Ma, & van der Aalst, 2013). Conformance checking thus aims to compare

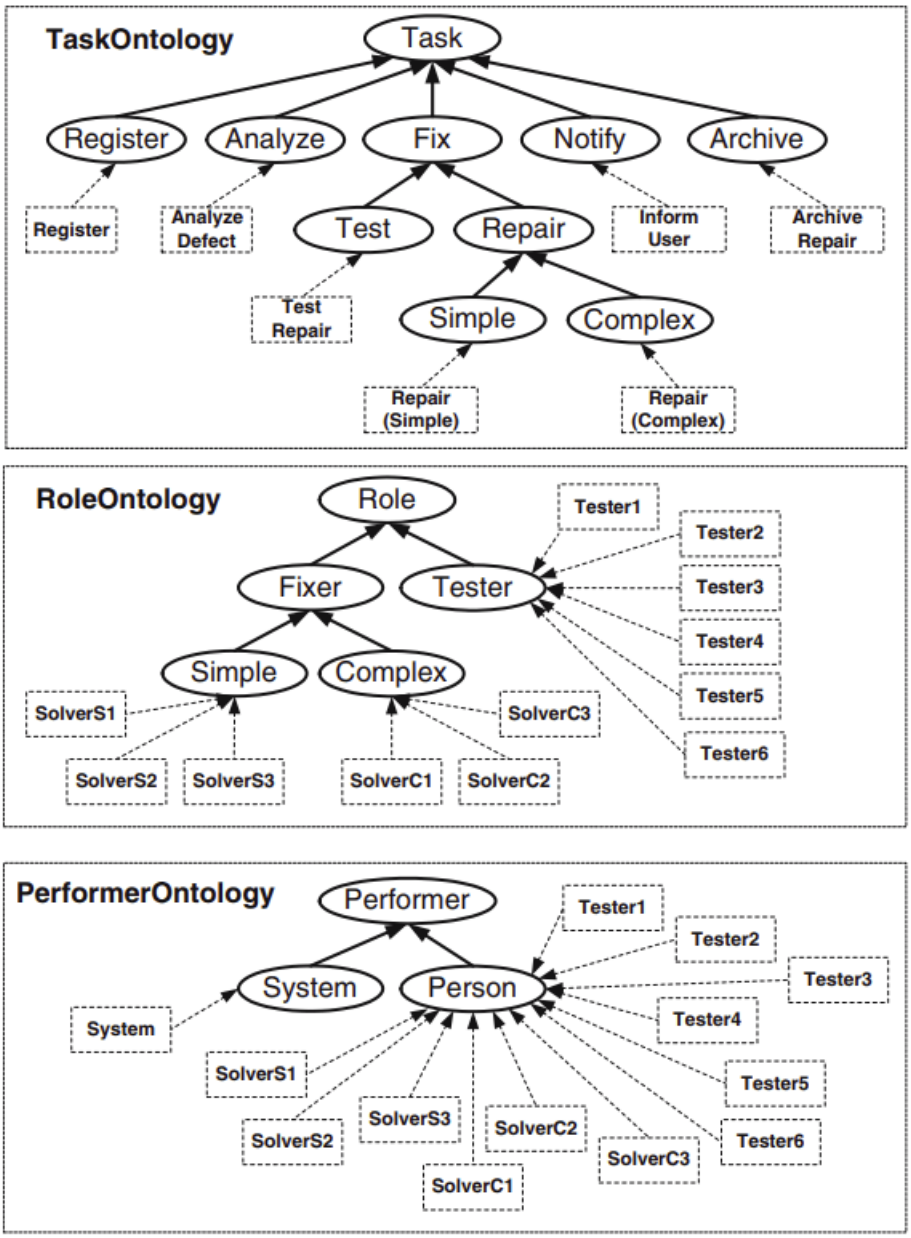



Figure 8: Example of process ontologies.(dos Santos Garcia et al., 2019)

and measure processes along these CMFs.

### 2.2.3 Process Enhancement and Predictive Techniques

Thirdly, there is the process mining type of *process enhancement*. This type of process mining focuses on enhancing the process model with information from outside the event log. The incorporation of relevant extra information can be combined with the process model using statistical and machine learning techniques to generate predictions(dos Santos Garcia et al., 2019) or offer different perspectives. These extended models and their associated predictions can be used to provide operational support, the most ambitious process mining form, and providing a powerful tool for data scientists(Van Der Aalst, 2016).

Linking process enhancement back to the c-level questions to be answered in Figure 7, de Medeiros and van der Aalst suggest that the answers to questions around topics like average-case throughput time; decision probabilities; critical activity, or resource identification; and (estimated) activity duration(De Medeiros & van der Aalst, 2008) can be answered when supported by process enhancement(dos Santos Garcia et al., 2019).

While not all too obvious, predictive methods and techniques can be considered to fall under process enhancement. These methods often require enhanced event logs containing more than just the basics of the case, activity, timestamp. As such, predictive techniques aim to use as much of the data available to make predictions about cases and activities.

Process mining has been combined with different predictive methods when trying to answer several of the c-level questions that can be asked, like instance duration, delay identification, and recommended execution path. These methods include but are not limited to, Bayesian indicators, "decision trees, case based-reasoning, recommender systems, and neural networks"(dos Santos Garcia et al., 2019). In 2014, Polato et al. were able to use the Naive Bayes principle combined with a Support Vector Regressor to predict both case path as well as estimate case duration(Polato, Sperduti, Burattin, & de Leoni, 2014).

In the end, of course, we would like to use these techniques to be able to provide real-time decision support, allowing for better business agility. A general application of predictive process mining is to provide suggestions in real-time to help reduce risk. In 2017, Marquez-Chamorro et al. suggested applying an evolutionary algorithm combined with process mining to provide active business process predictions(Márquez-Chamorro, Resinas, Ruiz-Cortés, & Toro, 2017). The evolutionary algorithm used is the genetic algorithm. This is akin to Darwinian evolutionary concepts in which over a random probability distribution iterative population-based optimization using a fitness function is performed to, over the course of several (simulated) 'generations', allow the 'fittest' of the population to remain (survival of the fittest).

In the same year, Evermann et al. suggested using deep learning to predict activity sequence and case behavior in a business process(Evermann, Rehse, & Fettke, 2017). By using recurrent neural networks combined with natural language processing, a process model did not need to be created. This avoids the (sometimes difficult) task of diving into the model quality metrics debate, making this option ideal if a model is difficult to generate(dos Santos Garcia et al., 2019).

Active prediction of the next activity was done by Lakshmanan et al. in 2013 by using decision tree learning methods while also providing an estimated degree of accuracy(Lakshmanan, Shamsi, Doganata, Unuvar, & Khalaf, 2015). The technique suggested using instance-specific probabilistic process models to discover the likelihood of given case sequential activities and outcomes. By transforming the probabilistic process model into a Markov chain, future tasks to execute can be predicted with higher accuracy than conditional probability methods(Lakshmanan et al., 2015). This technique allows cases to be viewed in parallel.

When trying to monitor and predict cases in active real life situations, it can be difficult to be able to view each case individually and as disjunct. Many running cases in the real world will be dependant on other cases being executed in parallel. To tackle this problem Senderovich, Francescomarino and Maggi proposed a framework in 2019 that analyzes the cases over a two-dimensional state-space of intra-case dependencies (information about only the case) and inter-case features (properties linking cases) for process prediction(Senderovich, Di Francescomarino, & Maggi, 2019). The latter encodes features using either a knowledge-driven or data-driven encoding approach. A derivation function is also used to reduce the dimensionality of the feature space - focusing only on the relevant $N$-features, avoiding the dreaded curse of dimensionality. This allowed for a data-driven approach, using an alteration of the K-Nearest Neighbor technique to estimate the 'distance' between cases. Senderovich et al. showed that their approach works when predicting actual execution-time and that a data-driven encoding approach "has a better or comparable accuracy" when compared to a knowledge-driven encoding approach, despite having no prior knowledge of the case type.

In 2017, Yang et al. took process prediction one step further by providing a data-drive process recommendation framework using a regression model(Yang et al., 2017). Their suggested framework consists of two broad phases: process analysis and process recommendation, the former of which is subsequently divided into its own respective sub-phases. Process analysis consists of "(1) clustering of historic traces based on similarity; (2) determining the cluster prototypes that represent the established process enactment for each cluster; (3) regression analysis to explore the correlation between cluster membership and context attributes; and (4) interactive visualization and statistical analysis of process traces"(Yang et al., 2017). The emphasis of this framework centers around the proper clustering of cases. This is done by using their proposed Trace Similarity based on Time Wrapping, which looks at the time performance of each case over its activities, and Exemplar-based Clustering using Hierarchical Clustering, Affinity Propagation and Density-Peak based Clustering. Cluster prototypes are generated by (1) "discovering the time-warped prototype using time warping paired with a divide-and-conquer strategy"; (2) "unwarping the timeline to find the prototype" ; and (3) "filtering and repairing the prototype for easier interpretation"(Yang et al., 2017). Finally, a regression model is trained to identify which cluster a case is most likely to belong to.

Using this regression model, Yang et al. moved on to the recommendation phase of their framework. In this phase, the regression model is used to define to which cluster a new case belongs. Having identified a cluster, the recommendation system suggests using the prototype process belonging to that cluster. While not mentioned by Yang et al., this type of process recommendation can be used to estimate total case time execution based on the created data-driven process prototype.

### 2.2.4 Process Mining Support Areas

As stated above, dos Santos Garcia et al. added a fourth type of process mining category to van der Aalst et al.'s initially defined three(dos Santos Garcia et al., 2019; Van der Aalst et al., 2012). This fourth category is support areas for process mining which acts as a kind of supplemental adhesive between all the categories by including elements like process mining applications, methods for process mining projects, architecture and tools, background, gathering, and cleaning of event logs and the previously mentioned ontologies of process mining(dos Santos Garcia et al., 2019).

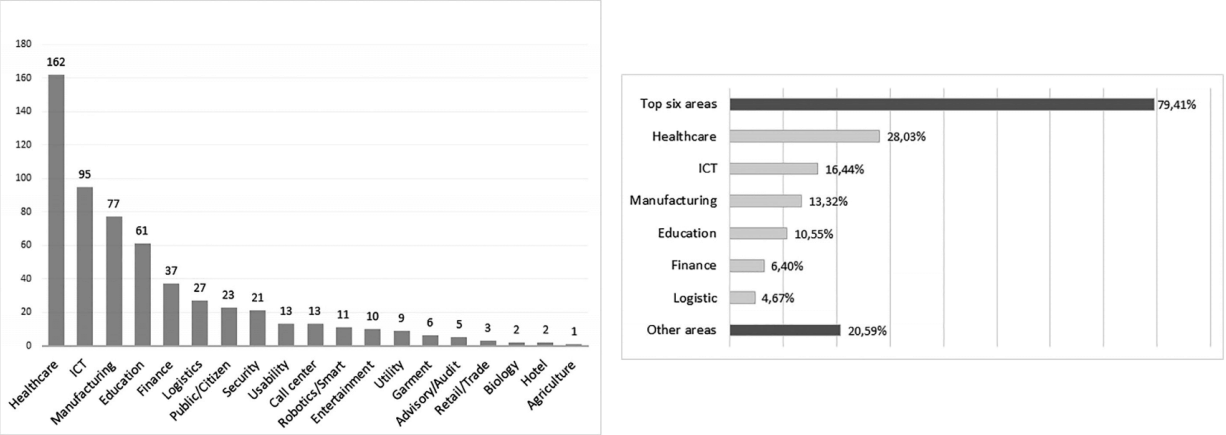



Figure 9: Number of process mining papers by application domain in 2019 (dos Santos Garcia et al., 2019)

Process mining has been applied in many different fields, the most prevalent of which are fields like Healthcare, ICT, and Manufacturing - with Healthcare taking the lead by far (see Figure 9). With the growing popularity of process mining and its application in many different fields, different architectures and tools have been developed and released to offer increased support in areas like Business Intelligence, Business Process Management, and Enterprise Performance Management. Most publications around process mining use the ProM framework despite alternatives being offered like Celonis Discovery, Disco from Fluxicon, Software AG, RapidProM(dos Santos Garcia et al., 2019), and recently the PM4PY Python package released by the Fraunhofer Institute for Applied Information Technology. These frameworks are further supported by a heterogenic data source format in the form of Extensible Event Stream (XES) files(X. W. Group et al., 2016).

For tackling process mining projects and combining it with other techniques, methods for process mining projects have also been suggested. This starts, again, with our process mining protagonist van der Aalst back in 2011. Van der Aalst suggested using the five staged $L^*$ life-cycle model when applying process mining to a project(Van Der Aalst, 2011). These five stages are structured as follows:

1. Plan and justify: this is where you ask the C-level questions you will likely explore.
2. Extract and explore the knowledge available, collecting and storing event logs.
3. Discover a process model using the previously collected event logs and the chosen mining algorithm. After having created a presentable model, perform conformance checks, and analyze

the activities and deviations to the expected model.

4. Use process enhancement to extend the model and gain new perspectives and insights.

5. Operational support: detect and predict unwanted workflows and provide recommendations. "This is the most advanced level of computational support, for example, the process mining tool should be capable to alert (email) on deviation cases, provide advice about bottlenecks, recommend resource setup or reallocation, ect."(dos Santos Garcia et al., 2019).

The data gathered during the extract (and explore) stage of van der Aalst's proposed application of the $L^*$ life cycle to process mining can come from many different sources. This brings with it fundamental issues around data quality. Event logs can have many issues including missing events; incorrect or imprecise timestamps; imprecise activity names; and irrelevant events(Bose, Mans, & van der Aalst, 2013). Ways of improving data (and thus event logs) quality can vary. These include identification and solving of imperfections, like misspelled activity names or different date-time formats(Suriadi, Andrews, ter Hofstede, & Wynn, 2017), or tracking different abstraction levels of activities in the process(Baier, Mendling, & Weske, 2014). The guiding principle around data quality and handling data in process mining is that it should be treated like a first class citizen(Van der Aalst et al., 2012).

Treating data like a first class citizen has become another one of those buzz terms, but it is quintessential to understand what that essentially means. These first class citizens are to be considered entities or objects in the computer science sense. In the 1960s Robin Popplestone already defined first class items as having fundamental rights: being able to be passed as parameters to a function; being able to be returned as a result of a function; being able to be subject of an assignment statement; and being able to be tested for equality(Popplestone, 1968). These items or objects can perform actions but they remain their own object. The more modern colloquial definition of data as a first class citizen refers to making it central to the process being analyzed instead of keeping it as a 'by product' for debugging or profiling(Van der Aalst et al., 2012). Treating data (in the objects it is presented) as a first class citizen has scientific benefits: "(i) to allow assessment of the reproducibility of results; (ii) to be reused by the same or by a different scientist; (iii) to be repurposed for other goals than those for which it was originally built; (iv) to validate the method that led to a new scientific insight; (v) to serve as live-tutorials, exposing how to take advantage of existing data infrastructure, etc."(Belhajjame et al., 2012) By storing the data in the form of first class objects that can perform actions as well as have actions performed on them, the data becomes central in the analysis of the process, preventing it from becoming a 'by product'.

## 2.3 Process mining in agile software development

As discussed in Section 2.2.4, process mining in the ICT domain is one of the most written about research areas in process mining. While the ICT domain is extremely broad, one of its subdomains that have drawn interest over the years is that of process mining in agile software development. This in itself can also be split in one of two directions: process mining of user actions to help software development teams improve the system or product being developed; and process mining of the actions software development team members take to analyze, identify and improve the processes team members go through when developing software.

In 2014, Rubin, Lomazova, and (of course) van der Aalst published a paper discussing the former (Rubin, Lomazova, & Aalst, 2014). The specification and design of (large) systems is a challenge, leading to designers often trying to anticipate possible software changes and working in a prescriptive top-down manner, reacting to feedback rather than predicting it. Rubin et al. would argue that this does not make ideal use of modern agile development methodologies which can enhance the development process by adding a bottom-up perspective to the design process. Agile development methods, which ever you choose, generally have one thing in common: iterate often over a minimal viable products while integrating active user feedback after each iteration. By creating event logs of user actions, the development team can integrate process mining into the design process. This allows the designers to combine a prescriptive top-down approach with a data-drive bottom-up analysis of real user and system run-time behavior, using user behavior to improve design and usability(Rubin et al., 2014).

While an interesting area of research, for this paper we will be focusing on the alternative direction: process mining of development teams to help identify and improve team processes during development. Using process mining, teams can identify the actual (agile) processes they go through while identifying unwanted workflows and improving agility(Erdem, Demirörs, & Rabhi, 2018).

In 2014, Mittal and Sureka used process mining to analyze the software development life-cycle of 19 undergraduate student teams taking part in a software engineering course(Mittal & Sureka, 2014). Data was collected from three different software repositories, reflecting three general stages of development the students went through for their course: team wiki for the requirements engineering stage; version control system for the development and maintenance stage; and issue tracking during the corrective and adaptive maintenance stage. All of this data was placed in a SQL database and processed from there looking into Process Verification and Conformance, Development Activity, Actual Process Discovery, Team-Work Collaboration, and Product-Process Correlation. In this way, Mittal and Sureka aimed to gain insights into the "degree of individual contributions, quality of commit messages, intensity and consistency of commit [activity], bug fixing [processes and quality], component and developer entropy, process compliance and verification"(Mittal & Sureka, 2014). This was done using visualizations, like radar charts of team member contribution and line graphs of the number of commits over time; metrics, like mean repair time of bugs; and algorithms, like the Fuzzy Miner for process discovery and comparing estimated with actual execution time. These visualizations of data shed light on unequal work distributions, commit patterns, and crunch before a deadline. Mittal and Sureka end by pointing out a crucial element to take into account when assessing individual contribution to projects: "While the metrics and visualization proposed are useful for the instructor, they should be handled with care when used for grading, because they can be easily tricked by redundant or needless commits or issues (bug reports)"(Mittal & Sureka, 2014).

Several years later, in 2017, Erdem and Demirörs published a case study looking into the applicability of process mining as a process management technique for an agile software development team using the Scrum methodology(Erdem & Demirörs, 2017). The team being looked into defined their workflow which was then compared with their actual workflow using process discovery and conformance checking. It became clear that definitions for 'Done' cases were not being upheld, reflecting a misinterpretation of the Scrum method, providing an improvement opportunity. One of the important notes to come out of the research done by Erdem and Demirörs is that event logs should be treated as first-class citizens which will allows for higher quality process mining analysis.

A year later, Erdem and Demirörs continued their work with Rabhi in a mapping study of process mining in agile software development(Erdem et al., 2018). In this study Erdem et al. found that due to the lightweight and people centric elements of agile approaches to software development, development processes are not formalized, leading to development teams defining their own sequential event process. This can cause inconsistency, instability, and unpredictability of the development process. Erdem et al. would go on to mention that agile methodologies like Scrum and Extreme Programming can help alleviate this problem by providing a prescription to achieve agility.

In the same year as Erdem et al.'s mapping study, Marques, Mira da Silva, and Ferreira published a case study looking into process mining the Jira (agile project management software) data over two projects of an IT organization using the agile Scrum methodology(Marques, da Silva, & Ferreira, 2018). Marques et al. approached the event logs from the case, control-flow, performance, and organizational perspectives. This allowed them to identify the most common issue paths, the issue paths with the longest average turnover, and the team member roles and activity types. While providing a fair amount of insight, this is, of course, limited as can be noticed by the inability to determine if stories are actually closed at the end of a sprint and some central Scrum activities not being recorded in Jira. Marques et al. conclude by emphasising that process mining can be used to analyze the previously difficult to measure implementation and processes of agile teams.

The following year, Caldeira and Cardoso published a study looking into using process mining to evaluate software development teams' efficiency(Caldeira, e Abreu, Reis, & Cardoso, 2019). In this study, Java programmer teams were assigned the same software quality task to be performed using the Eclipse IDE. The different actions performed in Eclipse while performing this task were recorded and stored in an event log. This event log was then mined to "discover development process models, evaluate their quality and compare variants against a reference model used as 'best practice'"(Caldeira et al., 2019). Caldeira and Cardoso found that teams with a complex process model were less effective and vice-versa. The more complex a process model became, the more spurious actions were being executed, reflecting less focused teams. In this way, process discovery and conformance checking could be used to measure team effectivity. Caldeira and Cardoso wrap up by stating that making team members self-aware of their processes helps improve process effectiveness and that identifying corrective actions could lead to better deliverables.

As can already be inferred, a quintessential part of process mining is directed at big data analysis of data that can be collected around the team or workflow being analyzed. This is all but explicitly mentioned when it comes to the extract and explore knowledge stage of the $L^*$ life-cycle. In 2021 Biesialska et al. published a systematic mapping looking at the use of big data analysis in agile software development(Biesialska, Franch, & Muntés-Mulero, 2021) by looking at over 350 studies spanning different publication sectors and continents, analyzing 88 of them. This systematic mapping shows data-driven software development in the literature directs itself at five general fields: "code repository analytics, defects/bug fixing, testing, project management analytics, and application usage analytics"(Biesialska et al., 2021). Regression, classification, clustering and optimization were the four types of problems and a fourth type of business analytics category were identified. The fourth BA category is adaptive analytics in which environments automatically adapt to changes to improve ongoing learning (think reinforcement learning).

BA insights are heavily dependent on the target metrics used. Discovering metrics with which to analyze data collected can be difficult. Olszewskatelal et al. suggested four areas of interest,

each with two quantitative metrics, for measuring large-scale agile transformations in software development organizations(Olszewska, Heidenberg, Weijola, Mikkonen, & Porres, 2016). These metrics are Request Journey Interval (customer service request lead-time), Process Interval (feature lead-time), Hustle Metric (resource, like money, spent), Business Value metric (number of releases per time period), Pacemaker Metric (continuous integration commit frequency), Bottleneck Gauge (responsiveness of feature handover in organization), Snag Metric (number of external trouble reports/bugs/incidents), and finally, Typical Snag Metric (average number of days externally trouble reports remain unresolved). These metrics are presented in Figure 10.

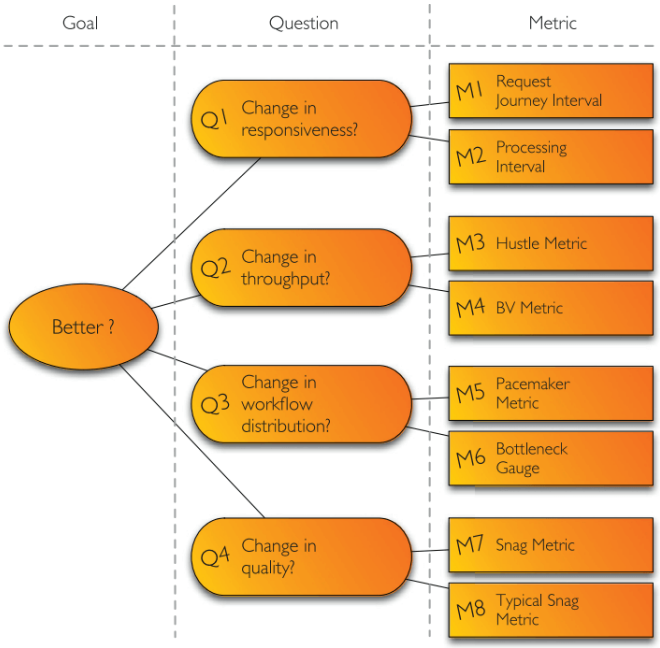



Figure 10: Quantitative metrics for measuring large-scale agile transformation in software organization(Olszewska et al., 2016)

Building upon the works of Olszewskatelal et al. and others, Boon and Stettina looked at using Jira backlog data to guide agile transformations with the aim of imporving organizational performance. By analysing Jira data over eight agile release trains, Boon and Stettina were able to provide a proof of concept on measuring agile transformation impact using Jira data, provide empirical assessments of transformations at FinOrg, and finally, compare their results with contemporary literature(Boon & Stettina, 2022). In providing a proof of concept, a table relating transformation objects, FinOrg transformation results, quantitative metrics, impact dimensions and the balanced scorecard (proposed back in 1992 by Kaplan and Norton(Kaplan & Norton, 2021)) was created and is presented in Figure 11.

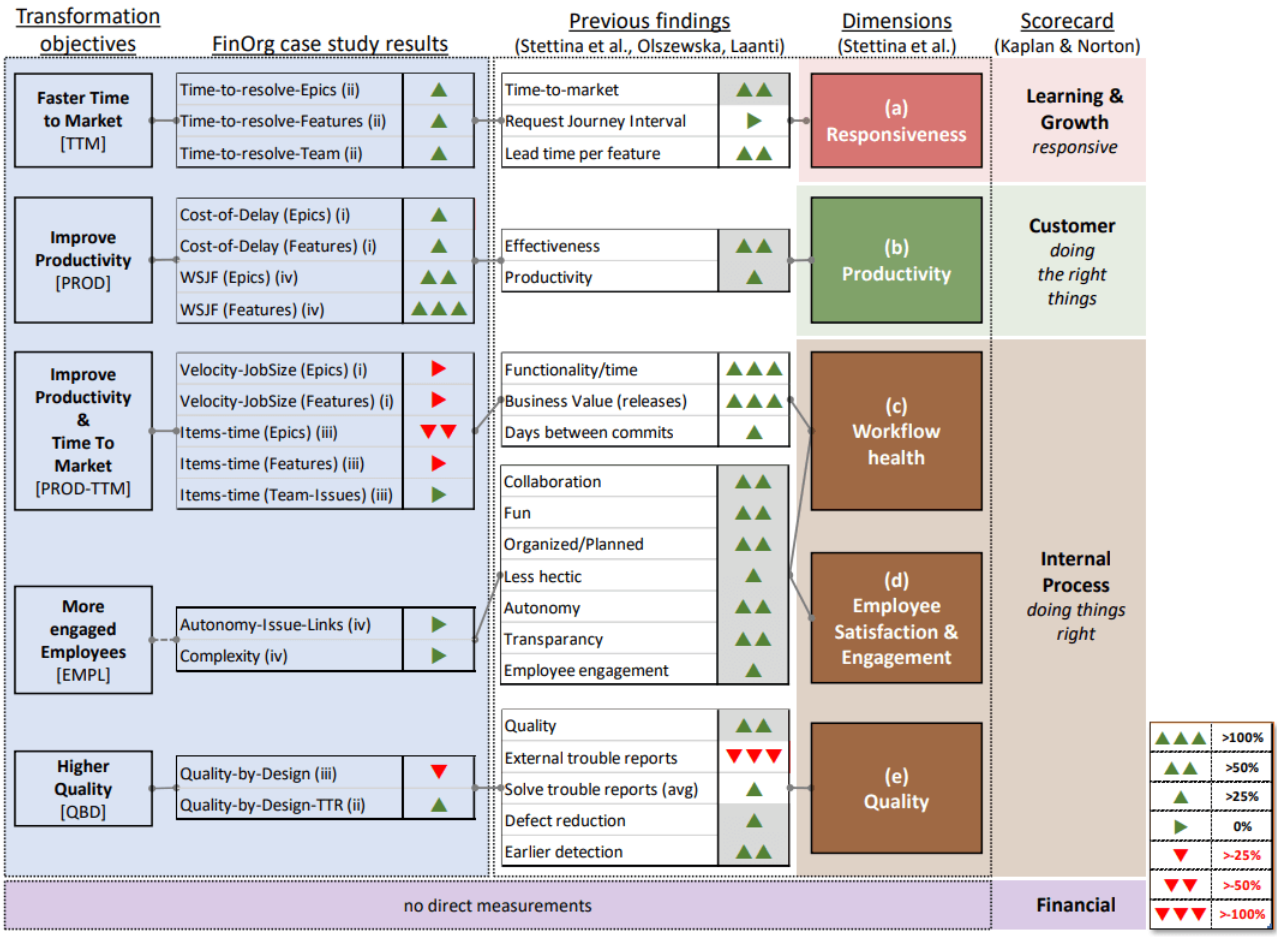



Figure 11: "Insights in how measures, objectives and perspectives are linked by establishing a connection between the Balanced Scorecard, the impact dimensions, and the measurements conducted during the transformation at FinOrg."(Boon & Stettina, 2022)

## 2.4 Literature review summary and gap analysis

This Section aims to summarize the entire literature review and fill in some of the gaps between the elements discussed. To help guide this summarization, this subsection will be structured according to the sub-Research Questions that it will be answering.

*What kind of software development methodologies are used by software development teams?* As software projects grow in complexity so do the seemingly unavoidable and inherent software development problems around cost overrun, time overrun, the software being unreliable, and the software failing to meet the users' needs or expectations(Davis et al., 1988). In an attempt to alleviate these issues, many software development life cycles and methodologies have been suggested since the 1970s. These methodologies can generally be split into heavy- and light-weight models, where the former is very planning and design-oriented - think waterfall - and best applied for projects where requirements are not likely to change and complexity is easily estimated. The latter, on the other hand, is used for projects where requirements are likely to change, complexity is difficult to estimate, and more iterative - think agile. Whatever the methodology, most software development

projects go through the stages of Research, planning, design, development, testing, feedback, setup, maintenance. Agile does so iteratively and has become the most popular development methodology.

A lot of different agile methodologies exist that all aim to emphasize customer involvement, iterative MVP development, self-organizing teams, adaption to change(Lau et al., 2009). Up to 32 agile practices have been defined in the Agile Manifesto(Williams, 2010), different subsets of which are used by different agile methodologies. The most popular agile methodologies are Scrum and an Extreme Programming-Scrum hybrid(Moniruzzaman & Hossain, 2013). Common agile practices include Story Cards, backlogs, epics, tasks, standups, Sprints, and team roles.

*What does process mining entail, what kinds of process mining exist, and what are the industry standards?* Process mining is a relatively young research field that has been growing over the last two decades(Van der Aalst, 2012b; Homayounfar, 2012). The main objective of process mining is to "discover, analyze, and improve business processes based on event data" (Van der Aalst, 2012a). In 2012, van der Aalst and colleges released a *Process Mining Manifesto* (Van der Aalst et al., 2012) defining terms and general process mining principles.
Several types of process mining have been defined with regard to input and output: process discovery, conformance, and enhancement. The data being analyzed consists of logs containing records of sequential *events*, where each event reflects a well-defined step (*activity*) in the process and is associated with a specific *case* (process instance)(Van der Aalst et al., 2012).

During process mining discovery, an event log is used to extract knowledge, identify sequential activities and develop a control-flow model without prior knowledge or information about the process(es). The actual processes teams use are revealed using only the behavior stored in the event logs(Van der Aalst, 2012b). Process discovery is done using miners, the most common of which are the Alpha and Heuristics miners (and variations)(dos Santos Garcia et al., 2019). There are four general perspectives to take into account when process mining: *control-flow*, *organizational*, *case* and *time* perspectives(Van der Aalst et al., 2012). These perspectives are very similar to process mining ontologies that aim to define the *what* of process mining.

With hundreds of different miners proposed over the years(dos Santos Garcia et al., 2019), choosing the correct process miner can be difficult. To be able to evaluate and compare the different miners, Rozinat et al. proposed a framework consisting of four quality dimensions(Rozinat et al., 2007): fitness, precision, generalization, and simplicity. These quality metrics allow different miners to be compared which each other and to see how well a mined model represents the event log data, which is very important during conformance checking.

With conformance checking the quality of a process model is measured along the quality metrics, applications of these measurements are defined and can then be used for active process monitoring. While often used interchangeably, conformance and compliance checking differ slightly. Conformance checking requires analysis of completed event logs after the project(s) is(/are) finished, while compliance checking is a form of monitoring using uncompleted event logs *during* run-time(Ly et al., 2015). *Compliance Monitoring Functionalities* - like reactively detecting and managing compliance violations; and supporting activity lifecycle - have been defined to assist in analyzing and exploring relevant ontologies(Ly et al., 2013).

Process enhancement focuses on extending the created model with information from outside the

event log. The incorporation of relevant extra information can be combined with the process model using statistics or machine learning to generate predictions(dos Santos Garcia et al., 2019) or offer different perspectives. Predictive techniques can consequently use the extended data models to make predictions about cases and activities. Predictive methods have been used to answer several of the c-level questions (see 7) that can be asked, like instance duration, delay identification, and recommended execution path. Commonly used predictive techniques include machine learning algorithms like Suppor Vector Regressors, Decision Tree Regressors, and Neural Networks.

Dos Santos Garcia et al. added a fourth type of process mining category to van der Aalst et al.'s initially defined three(dos Santos Garcia et al., 2019; Van der Aalst et al., 2012) which acts as a kind of supplemental adhesive between all the categories by including elements like process mining applications, process mining methods, tools, and cleaning of event logs and the previously mentioned ontologies of process mining(dos Santos Garcia et al., 2019). One of the process mining project methodologies to use is the five staged $L^*$ life-cycle model proposed by van der Aalst(Van Der Aalst, 2011), which divides the process up into: Plan and Justify; Extract and Explore; Discover Processes and Conformance Check; Process(/Model) enhancement; and Operational support (predictive and prescriptive).

*How is process mining used in predictive and prescriptive analytics?* Process mining can be used in predictive and prescriptive analytics to answer different c-level questions, like instance duration, delay identification, and recommended execution path. These methods include but are not limited to, Bayesian indicators, "decision trees, case based-reasoning, recommender systems, and neural networks"(dos Santos Garcia et al., 2019). These techniques can be applied to be able to provide real-time decision support, allowing for better business agility. Extending data models and applying machine learning techniques allow researchers and teams to detect and predict unwanted workflows and provide recommendations, for example in the form of alerts or emails(dos Santos Garcia et al., 2019). Depending on the machine learning technique applied, this method, combined with statistics, can be used to identify feature importance and effect on predictions, providing a prescriptive-like tool to help direct teams as they process unwanted workflows and try to find alternatives.

*How is process mining applied in (agile) software development?* Two general application areas of process mining in agile software development have been identified: process mining of user actions to help software development teams improve the system or product being developed; and process mining of the actions software development team members take to analyze, identify and improve the processes team members go through when developing software. Rubin et al. investigated the former where development teams can integrate process mining into the development process allowing the designers to combine a prescriptive top-down approach with a data-drive bottom-up analysis of real user and system run-time behavior, using user behavior to improve design and usability(Rubin et al., 2014). Mittal and Sureka, Erdem et al., and Caldeira and Cardoso have used process mining of development teams to help identify and improve team processes during development. Using process mining, teams can identify the actual (agile) processes they go through, while identifying unwanted workflows and improving agility(Erdem et al., 2018).

# 3 Method

In this Section, the methodologies used to perform this research will be discussed. The goal of these methods will be to help answer the research question of this paper: *How can process mining be used to suggest Scrum improvements for an agile software development team, and what kind of architectural design is required to investigate and provide these recommendations?*

The nature of software development teams is inherently complex and social in nature. With process mining, the emphasis is put on team workflows, which inherently are complex, social phenomenon. To analyze this synthesis of events, which are often unique to different teams, a case study of an agile software development team's workflow data will be used. For a little under a year, I was an active member of the team that will be analyzed, throughout which I was able to conduct informal interviews and observe the workings of the team. This allows for a mixed methodological choice where quantitative data of the team over several years can be combined with information gathered during the informal interviews and observations over a longitudinal time horizon.

The goal of analyzing the workflow data intrinsically pushes the data collection and analysis of this research towards a process mining project. Van der Aalst (one of the most prominent writers about process mining) suggested that all process mining projects should follow the $L^*$ life-cycle model(Van Der Aalst, 2011), presented in Figure 12. This life-cycle aims to guide all process mining projects through the different types of process mining (discussed in Section 2) towards a system that can offer active operational support. This operational support is often provided through predictive and prescriptive analytics using machine learning techniques. For this reason, data collection and analysis will be done following the $L^*$ life-cycle model. This model will also be the basis for the architectural design for this research for the same reason. The structuring of Sections 4, 5 and 6 will reflect the $L^*$ life-cycle model structure. In Section 4 an architectural design towards applying process mining to analyze Jira workflow data is presented.

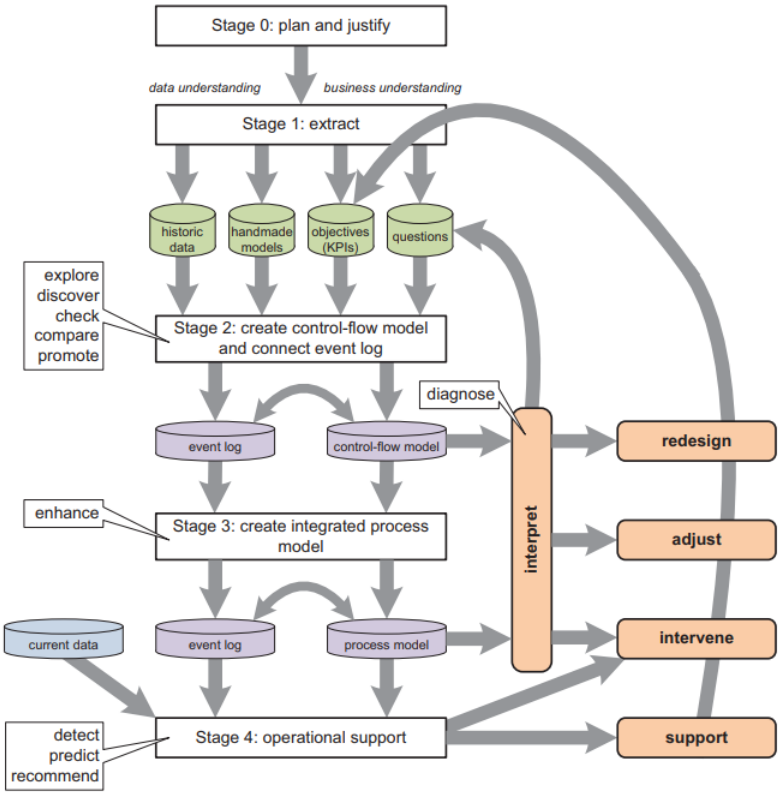



Figure 12: $L^*$ life-cycle model (Van Der Aalst, 2011)

### 3.1 Research strategy: case study

During this case study, I will be part of the software development team giving me an understanding of the workings and dynamics of the team. By way of this case study, I want to create a deeper understanding of the workflows the team goes through. This case study will use event log data, from a software development team using an agile development method, which will be collected, processed, and then analyzed using the different types of process mining: process discovery, conformance checking, and process enhancement with predictive techniques. The techniques used will be based on the techniques and best practices presented in Section 2. The resulting information will be incorporated into a predictive and prescriptive decision support system to allow for greater agility in agile software development by encouraging a data-centric approach to project management.

Data will be collected from the Connectivity .NET team at Indicia in Tilburg[1]. This team focuses mainly on developing APIs and other backend systems using Microsofts .NET Framework and .NET Core. This teamwork on large projects, including those that span several different teams, both internally within the company and externally. The team applies a Scrum methodology and uses Atlassian's Jira project management software. Jira allows teams to create projects, epics, tasks, set time estimations, set sprints, assign responsibilities, log time spent on a task in work logs, and visualize all properties over a Scrum board and other visualization tools. Jira also has an API that allows for easy and comprehensive data retrieval from projects.

[1]Indicia, Spoorlaan 348, 5038CC Tilburg. https://www.indicia.nl

The .NET team at Indica logged their first Jira sprint the first week of 2019. This means the team has been using Jira for a little over three years. From this team's Jira scrum board, data over 31 projects could be collected spanning these three years. Some of these projects are still in progress, while the rest are marked as completed. As many as 92 team members have contributed to events associated with data over the team's scrum board. As of writing, I have been part of this development team for a little less than a year.

The team uses an agile Scrum methodology to approach and complete projects. The team consists of developer roles (including a team lead), a Scrum Master (who also functions as a developer), and a (project/delivery) manager. Each day a stand-up occurs at 09:30 in which each team member defines what (s)he is working on and what was completed the previous day, and takes anywhere between 10 minutes and a half-hour. Tasks are generally created as needed and often consist of a summary describing the problem or what to do and done criteria are not explicitly defined by the creator of the task but can be inferred. No planning-poker occurs. Each sprint consists of two weeks and starts with a retrospective of the previous sprint. Team members often work on different projects simultaneously, where each project is assigned two or more team members. The team uses GitLab for version control and implements CI/CD through GitLab. Recent projects have been extended with unit tests to help this process. Along with unit tests, the team has applied a rule of testing the code during the review stage but is not always applied.

## 3.2 Data collection and data analysis: L* life-cycle

Using the above-mentioned approach, process mining will be used to identify actual processes the agile software development team goes through, identifying unwanted workflows and using suggested machine learning techniques to identify these unwanted workflows early in and during development, improving team agility.

To help guide this project, the $L^*$ life-cycle model will be applied as suggested by van der Aalst(Van Der Aalst, 2011) and discussed in Section 2.2.4. The structuring of this sub-section will thus reflect the $L^*$ life-cycle model and as such define the methodology behind data collection and analysis. In Section 4, an architectural design for applying a $L^*$ life-cycle model towards process mining of Jira workflow data will be presented. Sections 5 and 6 will also reflect this life-cycle structure.

### 3.2.1 Plan and justify

Data will not be sampled but rather all Jira issue data will be imported and stored in a local database. This data will then be analyzed using process mining and machine learning techniques to see if it can be used to provide both predictive and prescriptive analysis. Because this will mainly be a process mining exercise, the entire methodology and experimentation will be structured according to the $L^*$ life-cycle model, as discussed above. To assist in the planning and development of a Python application that can be used to perform this kind of research, an architectural design following the $L^*$ model will be designed and presented in Section 4.

The importance of treating data as a first-class citizen was emphasized by van der Aalst et al. in

the Process Mining Manifesto(Van der Aalst et al., 2012) and again in the results of Erdem and Demirörs' process mining study of a development team using Scrum(Erdem & Demirörs, 2017). Both of these cases are discussed in Section 2.3. To ensure that data is treated as a first-class citizen, as defined in Section 2.2.4, the data collected from the agile software development team will be structured into clearly defined entities, all code and UML diagrams will be publically available to allow reuse, and this paper should serve as an example or tutorial to allow other researchers to reuse the infrastructure created. The collection, processing, eventual decision support system will all be built-in Python. The code of this project will be available on Github[2].

This research project can be justified as an attempt to make process mining more accessible and less daunting to apply to actual software development teams in the field. By creating an architectural design, researchers can use this to help guide and structure their efforts. Furthermore, this paper aims to present a comparison between popularly applied process mining and machine learning techniques presented in the literature (see Section 2).

### 3.2.2 Extract and explore knowledge: collecting and storing event logs

Using the Jira API several different collections of entities and their associated attributes can be retrieved. A collection of the projects, issues (tasks) with their associated changelogs, and team members are among the entity sets that can be collected. A connection to the Jira API will be made using the Python Jira package[3] and the entities will be managed and worked with using the Object Relationship Manager (ORM) elements of Python's SQL Alchemy[4].

As discussed in Section 2.2.4, it is important to treat our data as first-class citizens. For this reason, models reflecting the entity structure will be defined and used as their own respective class, being able to use the entity as a parameter, return value, in an assignment statement, and test it for equality. Furthermore, the infrastructure to be used should initially be designed and presented in unified modeling language (UML) models and Archimate to provide a clear and manageable infrastructure, thus allowing for (infrastructure) reproducibility and reuse. An architectural design follow the $L^*$ model will be presented and discussed in Section 4.

The data will be stored following the entity-relationship diagram (ERD) presented in Appendix C. In total 10 different entities will be stored and processed. These entities are Project; Issue (task); Issue Type; (issue) Priority; (issue) Status; Status Category; (issue) Time Tracking; (issue) Progress; (issue) Worklog Item; and Team Member; To be able to properly process the models of the entities, the from Jira collected values of these entities will be stored in a (local SQL) database and processed using the Python ORM SQL Alchemy.

Despite having collected a large amount of data from Jira, this data still needs to be structured in such a way as to form an event log and be ethically usable. Ethical implications are discussed further in Section 3.2.4. The Jira data is a collection of footprints and these footprints need to be structured in such a way as to provide meaningful information. Van der Aalst defined the contents of an event log as consisting of a collection of events reflecting sequential activities associated with

[2]Full details: https://github.com/akannangara/ProcessMiningThesis
[3]Python Jira package: https://pypi.org/project/jira/
[4]SQL Alchemy full details: https://www.sqlalchemy.org//

a case, and are often - but not always - accompanied by timestamps and resources(Van der Aalst, 2012a). This was discussed in Section 2.2. The most basic form of an event log, which will be applied, is a collection of activities and timestamps associated with a set of cases.

Having structured the data in such a way as to make it reflect actual event logs, the file needs to be converted to the heterogenic data source format that is generally used in process mining. At this point, the event logs data will still be structured as entities or objects in Python. As discussed by Group et al.(X. W. Group et al., 2016), most process mining frameworks are supported by the XES file format, a data source file format designed to help with event-based data. Many frameworks, including ProM or PM4PY, also accept CSV files. For simplicity, the event log data will be converted to CSV files before being used for the process mining steps itself.

### 3.2.3 Discover a process model: process discovery and conformance checking

Having developed a usable event log, the next steps will be to discover the actual processes the team goes through and compare them with the expected processes; process discovery followed by conformance checking. This will be the first real experiment performed during this research, where different miners building models of the event log are compared along with the quality metrics. This aims to partially answer the sub-research questions: *How do process mining techniques and standards compare?*

Given the type of data that can be collected with the Jira API, the focus will be put on how and when issues move across the Jira Scrum Board. This will reflect processes that can be defined by sequential activities over the Scrum Board and will fall into the *control-flow perspective* of process mining discovery. The different perspectives of process mining were discussed in Section 2.2.1.

The most commonly used process miners from the control-flow perspective in process discovery are the Heuristics Miner and Alpha miner (including variations)(dos Santos Garcia et al., 2019). Given that these are the most popular miners, and the Python process mining framework PM4PY supports them, both of these miners will be used. As a variation on the Alpha miner, the Alpha+ miner will be the third miner assessed. The inductive miner will also be used given its polynomial complexity and enforcement of soundness and fitness; and as it is also supported by the PM4PY framework.

These three different miners will then be compared and evaluated according to the four quality dimensions defined by Roziant et al.: fitness; precision; generalization; and simplicity (Rozinat et al., 2007). These were described in detail in Section 2.2.1.

The Heuristics and Inductive miners both have threshold parameters that have to be defined. For the Heuristics miner, the parameters ate the dependency threshold, length-one loops threshold, and the length-two loops threshold, which usually have the same value and are usually set to 0.9(A. Weijters & Ribeiro, 2011). Varying threshold values can have a profound influence on the quality of the generated model and as such should be considered before actual application(Suhendar, Wisudiawan, & Herdiani, 2018). The threshold values that will be considered for the heuristics miner are will vary between $(0.0, 1.0)$, and will be compared by Bayesian optimization using Gaussian Processes, aiming to minimize the inverse quality score (in other words: maximize quality score). The Inductive miner only has one threshold value, the accepted noise threshold value. While usually set to 0.0(for Applied

Information Technology, n.d.), the values $\{0.0, 0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ will be considered.

The most suitable process discovery miner for the given event log will be selected based on the average score of fitness, precision, generalization, and simplicity, and used when generating a process model for the desired scrum board control flow. Generated process models will be subject to process conformance and compliance checking (the difference was discussed in Section 2.2.2) when comparing the desired workflow with the actual event log. The team Scrum Master, Team Lead, and Manager will be asked to define the desired workflow for issues across the Scrum board. This desired workflow will be developed into a model using the most suitable process discovery miner (defined above) and compared to the actual event log by way of conformance checking using the quality metrics. The questionnaire used to collect the desired workflow is presented in Appendix F.

By comparing the event log with the desired workflow model, each issue can be assigned a fitness value. This value reflects how well the issue follows the desired workflow path (control-flow).

### 3.2.4 Use process enhancement: process enhancement by extending the model

During process enhancement we will extend the event logs with additional information, using this incorporated information combined with the developed process models to make predictions on the control flow of issues by way of machine learning or statistics. This can then be used to provide predictive and prescriptive analysis. Process enhancement is discussed in detail in Section 2.2.3. To stick with the $L^*$ stages structure, predictive techniques used to generate predictions will be discussed in the next subsection, Section 3.2.5, and as such this subsection will only handle model enhancement and not predictive techniques.

During this stage of the $L^*$ life-cycle, the methodology will be directed at extending the basic event log with more comprehensive information to gain more insight and be able to make predictions in the long run. The first enhancement was already discussed in the previous subsection, where each issue is assigned a fitness value as compared with the desired workflow. The event log will be converted to a data set to be used during the machine learning stage. A new entity model is created for this step to clearly define all of the fields to be analyzed in the next stage by way of machine learning techniques.

This is where ethical implications of the data come into play and have to be thoroughly considered. In her book *Weapons of Math Destruction*, Cathy O'Neil dives into how AI's might heavily base their decisions or suggestions on a piece of information that we as humans might not deem relevant or morally acceptable to take into account. For example, in a recidivism program used by judges in several states in the United States, race and gender were the most heavily weighted factors for the AI that would assess the recidivism chances of a convicted criminal(ONeil, 2016). This often stems from the misconception that AIs, if properly implemented, will consider the correct aspects. But the aspects any AI will take into account are all of the aspects provided.

Given that we will later be using different AI algorithms to help assess and identify the likelihood a task belongs to a certain subset of tasks, we have to look into the morality of the data being processed. Team members are one of the resources associated with a task, and one of the resources

that can easily be identified and singled out. To avoid singling out a team member in front of management, all team members will have to be anonymized *before* moving on to the next step and processing the newly generated event logs. The team members will be categorized into groups to be defined by the Team Lead or Manager. Any further steps should therefore not be processing the team member individually, but the team member category to avoid any predictive system focusing on a single team member.

Each issue status change (item of the event log and machine learning data set) will also be extended with inter-issue information. As noted in Section 2, and emphasized in the paper by Senderovich et al.(Senderovich et al., 2019), cases cannot be viewed individually or as disjunct. Issues will of course influence each other. If for example too many issues are planned in one sprint, the team might be overwhelmed and their productivity could be affected. To allow for inter-case information to be added to each data set item, items will be enhanced with the number of issues in the sprint, the sprint week number (numbered fortnight of the year, 1 to 26), and the sum of the sprints estimated time. These are all fields that are not automatically associated with an issue in Jira. For this reason, these fields have to be calculated. Active issues are difficult to identify in this way since non-active issues (not part of the sprint) can still be marked as 'To Do' instead of 'Backlog'. For this reason, the number of issues in the sprint and its time estimate should be considered more as rough estimates. The issues to view (with their status at the time) will be those that are not yet marked as completed before the end of the sprint, which have had a change registered in the changelog during that sprint, or which have been moved to that sprint. How often the sprint has changed since the issue was created and since the issue status was last changed is also registered as a variable in the data set.

The machine learning data set will be an extension of the event log containing only completed issues, being extended with an issue fitness value and the eventual work ratio of the completed issue. A class diagram containing a model of the machine learning data set item is presented in Figure 13. The computed fields in this diagram are values that cannot automatically be taken from the completed issue, as they might change while an issue is being worked on, and thus have to be discovered by scanning the issue changelog. An issue item is marked as changed (either since the last status change or since creation) if the issue priority, issue type, summary, description, attachment, assignee, or due date is changed. Values like priority or current status are inherently ordinal so only require assigning an integer value to these fields. Properties like issue type and assignee member are nominal in nature and as such need to be encoded using one-hot encoding. This machine learning data set will be stored as a CSV file, being used to try and make predictions and provide recommendations.

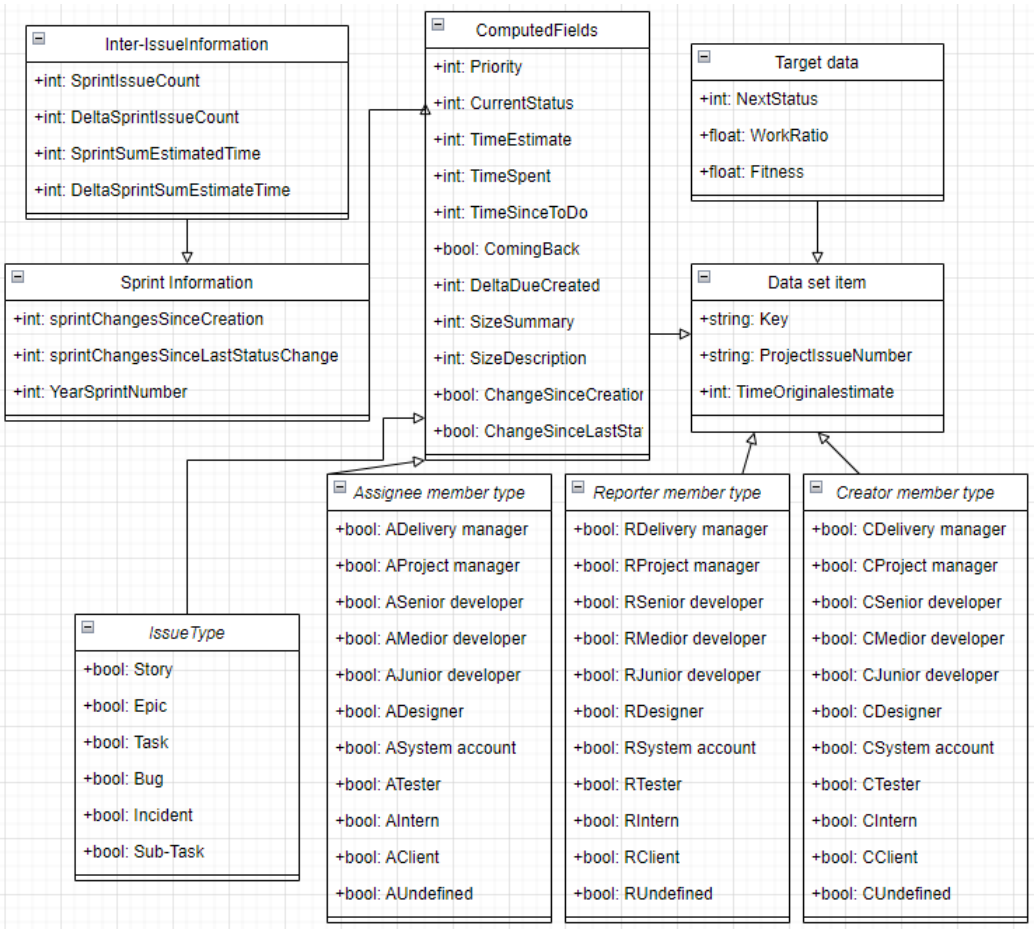



Figure 13: Class diagram of ML dataset. The computed fields in this diagram are values that cannot automatically be taken from the completed issue, as they might change while an issue is being worked on, and thus have to be discovered by scanning the issue changelog. An issue is marked as changed (either since the last status change or since creation) if the issue priority, issue type, summary, description, attachment, assignee, or due date is changed.

### 3.2.5 Operational support: apply predictive techniques to provide recommendations

In this section of the methodology, we will be using machine learning techniques on the extended event logs (machine learning data set) to try and identify and predict task fitness (as compared with the desired workflow), task work ratio, and next issue status as tasks move across the Jira Scrum board, and the next status the issue will take on the Scrum board. The application of machine learning techniques to generate predictions in process mining was discussed in Section 2.2.3.

The issue fitness values will have been identified during the conformance checking stage of the $L^*$ lifecycle model discussed in Section 3.2.3 and issues will have already had an assigned work ratio value as all issues in the data set are completed. In this case, either the fitness value, work ratio, or next issue status are the target values for the machine learning techniques.

Each (completed) Jira task has a work ratio value. This is calculated simply by

$WorkRatio = \frac{TimeSpent}{TimeEstimate} * 100$. By using this value as the target value, machine learning techniques can be used to estimate this value based on all of the properties provided in the machine learning dataset. In this step, it would be important not to include other target values or the issue ID as that should not be relevant when predicting the time-overrun of a newly created task. For all machine learning techniques, only the relevant target variable is included and the issue ID is excluded.

When estimating work-ratio and issue fitness with machine learning techniques, only the completed issues will be used in the data set. Completed issues are issues that have been marked either Done or Rejected. The data set will thus contain a little under eleven thousand issues each extended with their issue fitness and work ratio, and so around 27 thousand samples.

The next step is to define the machine learning algorithms to use to make the predictions for the two test cases. A lot of different machine learning techniques have been applied in recent years around process prediction, as discussed in Section 2.2.3, ranging from Bayesian indicators, decisions trees, and neural networks to Support Vector Machines and multiple regression. For simplicity, three algorithms will be selected to compare that can be used for both test cases.

Based on the literature discussed in Section 2.2.3, three machine learning techniques stand out as usable in process control-flow prediction while having comprehensive a Python package and support in the Python SciKit Learn package[5]. The first machine learning algorithm to use is decision trees, which were also applied by Lakshmanan et al. in their 2013 process mining study into the likelihood of case sequential activities and outcomes(Lakshmanan et al., 2015).

The second selected machine learning algorithm to use is a Support Vector Regressor, which was also applied by Polato et al. in their 2014 study estimating case duration(Polato et al., 2014). A support vector regressor is a subdomain of Support Vector Machines which unfortunately has a time complexity of more than quadratic the input. Due to the high time complexity, a linear support vector regressor is recommended for data sets larger than several tens of thousands of elements(*sklearn.svm.SVR*, n.d.). Our dataset has about eleven thousand completed issues, spanning 28000 data elements, so the Linear Support Vector Regressor will be used as our second machine learning technique.

The final machine learning algorithm to be used is neural networks. In 2017, Evermann et al. used deep learning to predict sequence and case behavior. Similarly, neural networks can be applied to predict the activity sequence, or rather associated issue control-flow, for a given case.

The accuracy of these three machine learning techniques will be compared over the entire extended event log for the three test cases - estimating fitness, estimating work ratio, and predicting the next issue status. To this end, the machine learning data set will be divided into train and test sets into the industry standard of 80% train and 20% test. For each of the total 9 experiments (three machine learning algorithms over 3 cases), the train and test set will be newly generated from the extended event log. Accuracy for each of the three algorithms over the three test cases will be measured using a $R^2$-score.

[5]SciKit Learn Python: https://scikit-learn.org/stable/index.html

Finally, for each of the three trained machine learning models, feature importance will be presented. Feature importance can be used to provide prescriptive analysis as these are the features to pay the most attention to. While this is easy for the SVR and Decision Tree Regressor models (these models in Scikit Learn have attributes that represent feature importance), it is extremely complicated for a black-box neural network. For the neural network, feature importance will be defined by comparing Pearson's R (regression) and mutual information(Kraskov, Stögbauer, & Grassberger, 2011) (regression) over the data set.
Each of the machine learning algorithms that will be applied has several collections of different hyperparameters that can be tuned to improve the performance of the machine learning pipeline. Finding the right combination of hyperparameter values can result in huge pipeline performance boosts(Yu & Zhu, 2020). Brute forcing hyperparameter tuning can be a long and exhaustive process, costing a lot of computing power and time.

Due to the relatively small number of features used to predict the target variables (60) more than half of which is generated by one-hot encoding, feature selection will not be applied before training or testing of the machine learning techniques.

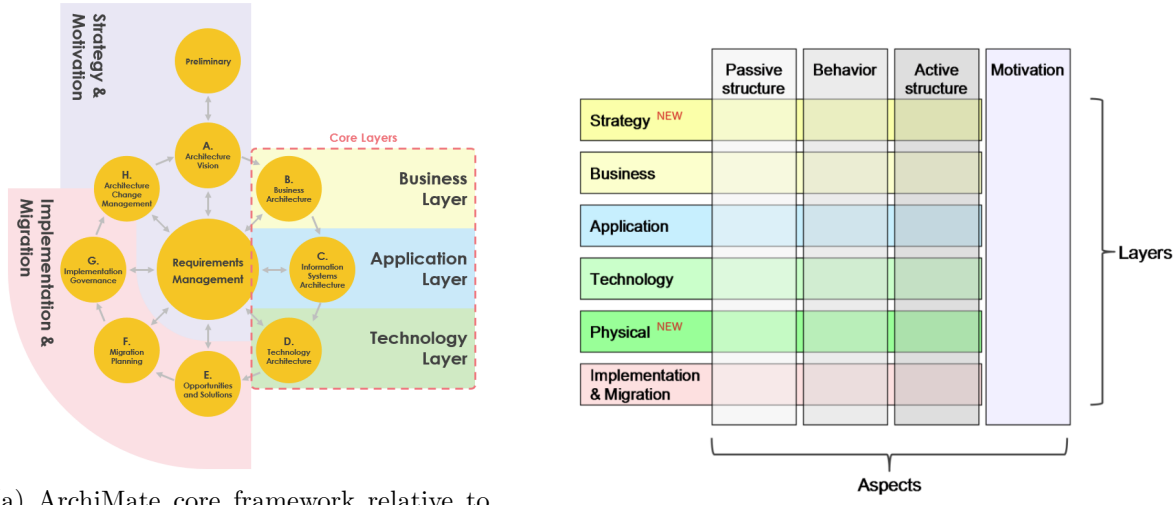
To avoid this tedious process, several techniques have been suggested to optimize the process of hyperparameter tuning. One of the suggested techniques to apply is Bayesian Optimization, proposed by Snoek et al. in 2012(Jasper Snoek & Adams, 2012) and at the time state-of-the-art. This same technique is used to define threshold values for the Heuristics Discovery Miner (see Section 3.2.3). Bayesian hyperparameter optimization was shown to achieve better results with fewer function evaluations than a standard-grid search over all hyperparameters or a random-grid search(Jones, 2001), outperforming other global optimization algorithms.

Despite more complex hyperparameter optimization algorithms having been developed since 2012 and bayesian optimization not being the standard, it will be applied in hyperparameter tuning due to its simplicity and ease of implementation within the Python Scikit Learn package, which is already used to implement the desired machine learning algorithms.

Using the trained machine learning models, by keeping all values constant at the mean or median (in the case of one-hot encoded features) values, the effect of the most important features on the predicted target variable can be shown. If the predictor is accurate, this technique can be used to identify how features affect the target variable and so identify areas of improvement for the team being analyzed.

# 4 An architectural design for process mining of Jira backlog data

In this Section, the architectural design for the experimentation will be presented and briefly discussed. The design will be created using The Open Group's enterprise architecture modeling language called ArchiMate and supported by Unified Modeling Language (UML) models. ArchiMate aims to provide a single language within which business processes, organizational structures, information flows, IT systems, and technical infrastructure can be defined(Visual-Paradigm, n.d.-b). The core framework defines three layers to any system, a business, application, and technology layer, which are then parsed over three aspects, a passive structure, a behavior aspect, and an active structure. These core layers make up part of the TOGAF (The Open Group Architectural Framework) model which is shown in Figure 14a. The complete ArchiMate framework as of version 3.0 extends the core elements and is presented in Figure 14b.




(a) ArchiMate core framework relative to the TOGAF framework(Visual-Paradigm, n.d.-b)

(b) ArchiMate 3.0 framework.(Josey et al., 2016)

Figure 14: ArchiMate framework

To assist in developing an architectural model, viewpoints can be used. Viewpoints are diagrams representing a part of the system architecture developed for a given purpose and directed at relevant stakeholders by representing the relationships between ArchiMate elements. In total, the ArchiMate 3.1 specifications define 23 unique viewpoints that can be used as a starting point when defining system architecture(T. O. Group, n.d.). These viewpoints are meant as a starting point and can be fused or extended where necessary. Four general viewpoint categories are defined: Basic, consisting of the core layers Business, Application, and Technology; Motivation, consisting of the motivational aspect across the system layers; Strategy, which presents a high-level description of enterprise strategy; and finally, Implementation and Migration, consisting of program-project relationships and architectural change management models(Visual-Paradigm, n.d.-a).

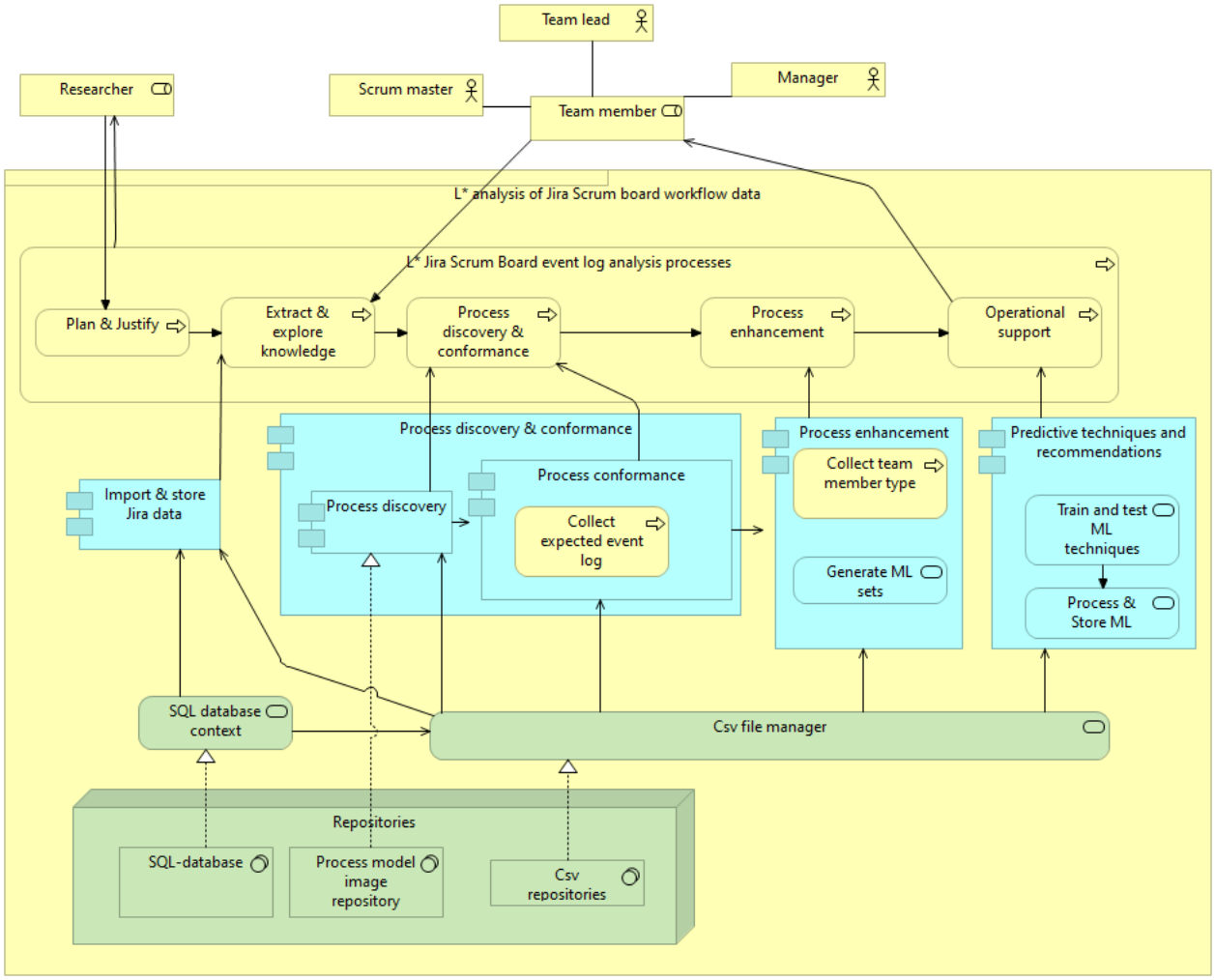



Figure 15: ArchiMate model of L* life-cycle approach to research from an extended Application Usage viewpoint

To define the architecture of the L* life-cycle approach to using process mining and machine learning to analyze Jira Scrum Board workflow data, the Application Usage viewpoint from the Basic viewpoints category will be used and extended. This viewpoint gives a broad number of potential stakeholders, including end-users, architects, and developers, while being used for system design and defining system composition and aggregation(T. O. Group, n.d.). The basic version of this viewpoint usually only consists of a business and application layer, showing the relationship between the two. To allow for a more complete view of the application being built to use during experimentation, the technology layer will also be included where needed.

A general overview of the system design in an ArchiMate model is presented in Figure 15. The design aims to show the system as a product focusing on the L* life-cycle of the process mining approach to this research. Two general business roles will be identified, that of the agile software development team member, consisting of a Team Lead, Scrum Master, and Manager, and that of the researcher who will be comparing and contrasting different process discovery miners and machine learning techniques.

The four application components shown in Figure 15 can be expanded into their own models using an extended version of the Implementation and Deployment viewpoint which focuses on designing for the application architects. These components are presented in Figures 16, 17, 20, and 21.

The first (Figure 16) shows the importation steps of collecting the Jira data and storing it in a local database. This database is then used to create an event log.

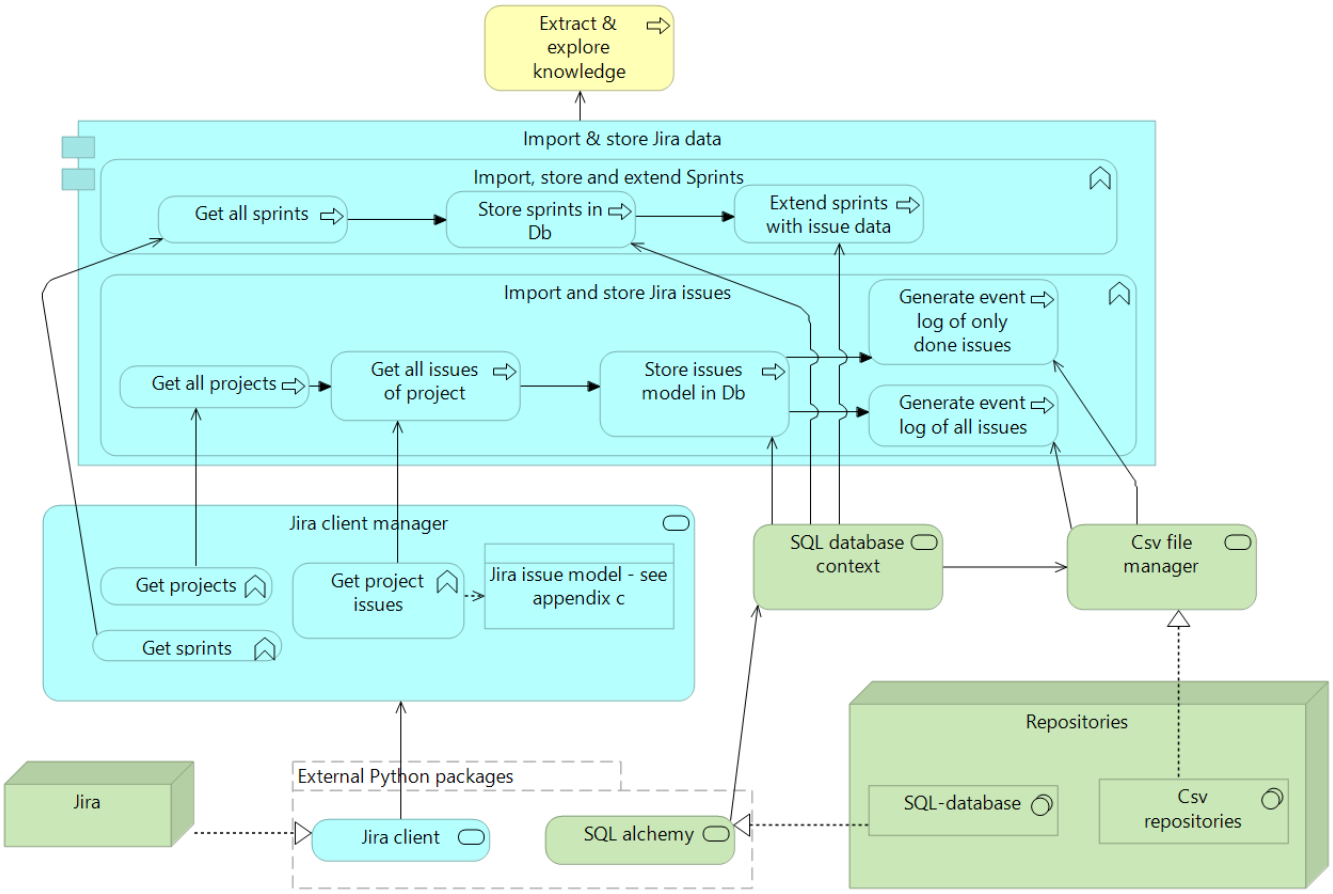



Figure 16: ArchiMate model Jira data importer from the Implementation and Deployment viewpoint

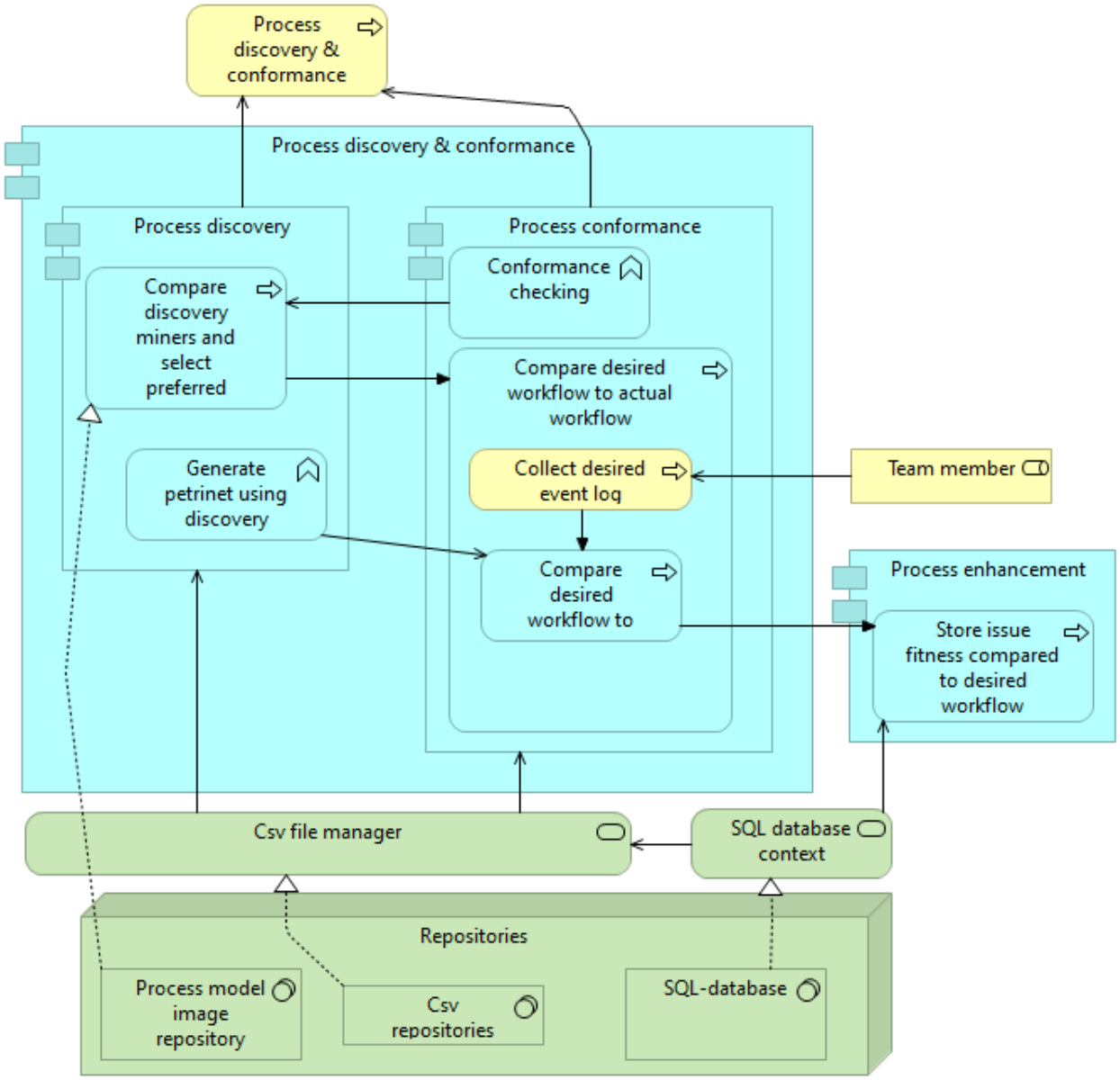



Figure 17: ArchiMate model process discovery and conformance checking from Implementation and Deployment viewpoint. This component is fairly large and can be divided into two small models. The component Process Enhancement is referenced simply because the fitness score of each issue as compared with the desired workflow is calculated by the Process Conformance component but stored by the Process Enhancement component.

The architecture of process discovery and conformance can be further split into two components, process discovery and conformance checking with the desired workflow. During process discovery, the Alpha, Alpha+, Heuristics, and Inductive miner experiments are conducted and their comparative results presented. Using the selected algorithm (with the selected threshold values), a model of the desired workflow can be created. The desired workflow is defined by the Team Lead, Scrum Master, and manager as discussed in Section 3.2.3. Architectural models for these two components

are presented respectfully in Figures 18 and 19.

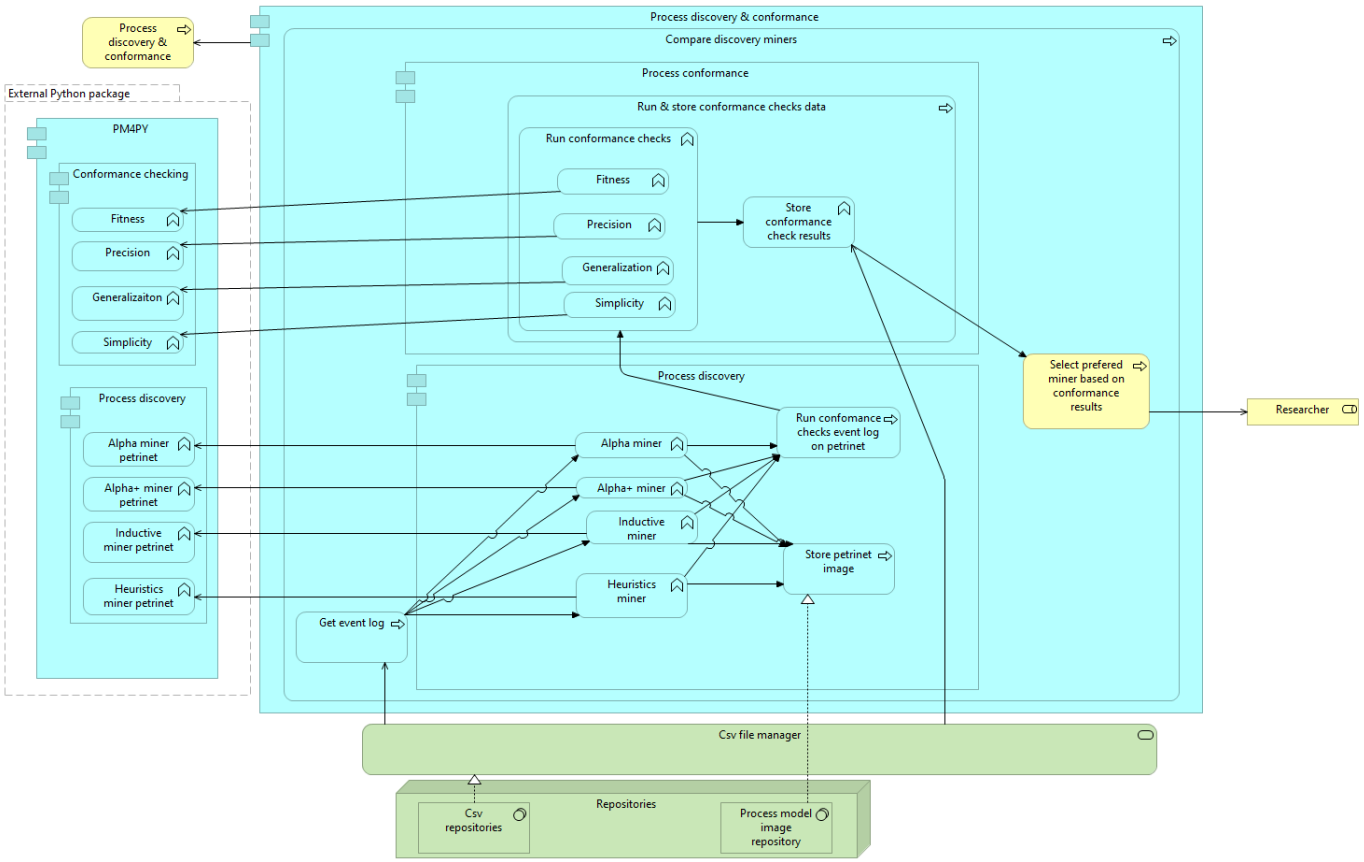



Figure 18: ArchiMate model of model discovery from the Implementation and Deployment viewpoint

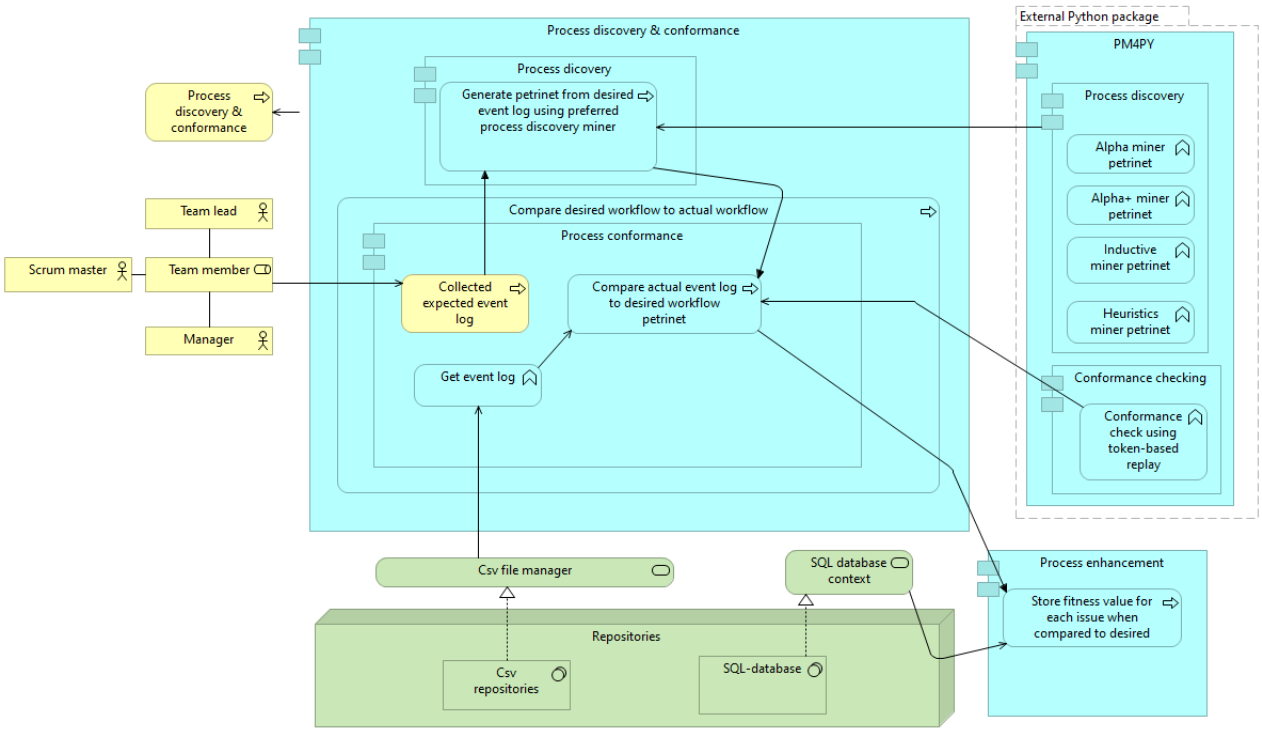



Figure 19: ArchiMate model of conformance checking compared with the desired workflow from the Implementation and Deployment viewpoint

Figure 20 presents the general workings of the process enhancement component. This shows how the fitness per issue is transferred to the process enhancement component, which adds the fitness element to each issue. The model also shows the steps of assigning each team member a team member type (discussed in detail in Section 3.2.4) and finally creating the machine learning data set.

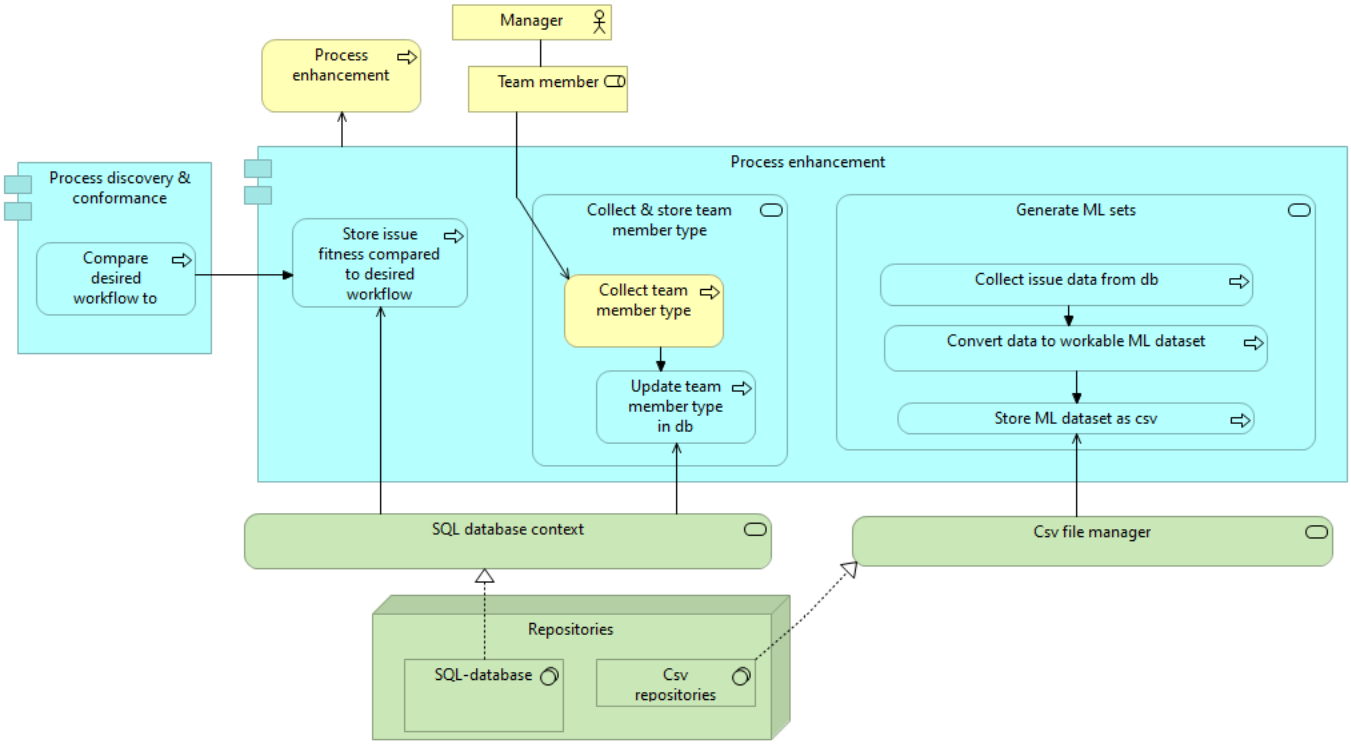



Figure 20: ArchiMate model of model enhancement from the Implementation and Deployment viewpoint

Finally, Figure 21 presents the Predictive Techniques component. The component uses the machine learning algorithms Multilayered Perceptron, Linear Support Vector Regressor, and a Decision Tree Regressor to try and predict issue fitness or work ratio using the data set created in the previous $L^*$ life-cycle stage. The hyperparameters of these machine learning algorithms are tuned using Bayesian optimization over a Gaussian process. The train and test data sets are the same size over the three algorithms (test size of 0.2).

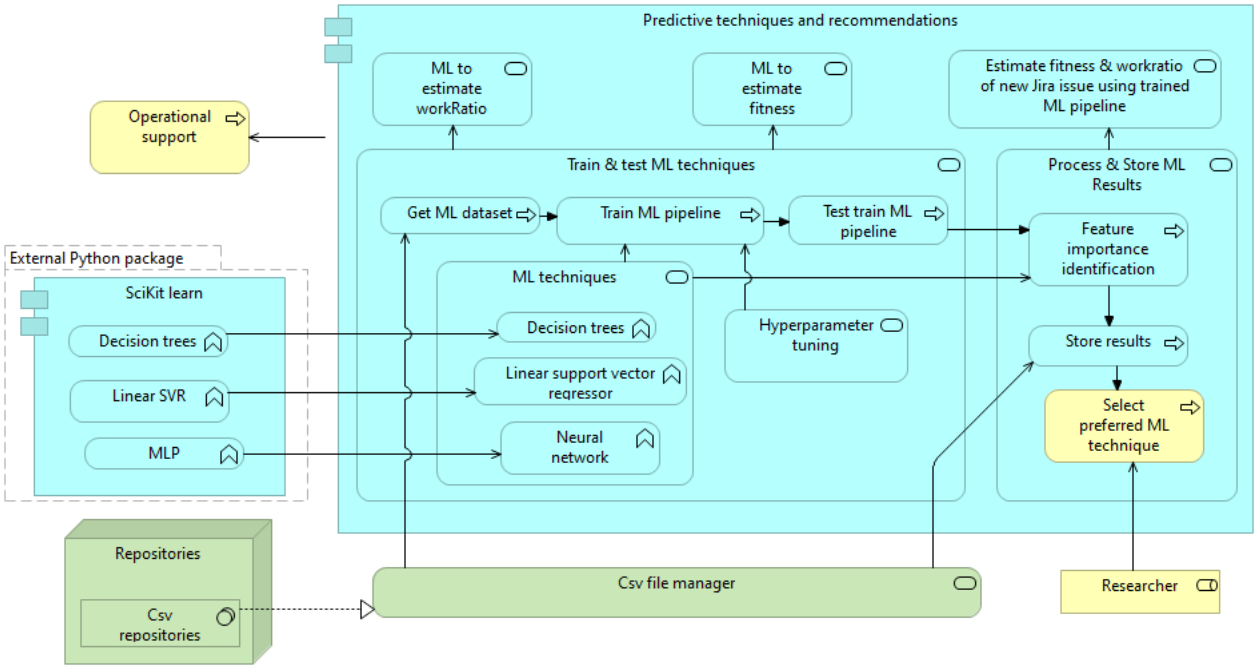



Figure 21: ArchiMate model predictive techniques from the Implementation and Deployment viewpoint

# 5 Experiments and results from implementing the process mining architecture

As discussed in Section 3, this Section looking into the experiments and results will be structured according to the L* life-cycle stages. This also reflects the structure followed by the experimentation application, designed and defined in Section 4.

The first stage of the L* life-cycle will be skipped as it was completely covered in Section 3 and then extended with the architecture models presented in Section 4. The second stage, Extract and Explore Knowledge, will be presented here but slightly awkwardly as it will only contain results summarizing the data collected from Jira. The fourth stage, Model Enhancement will also be presented in this Section despite not being part of an experiment due to its relevant role in the data extension process for creating a machine learning data set.

## 5.1 Extract and explore knowledge

This subsection will contain a summary of the data collected from Jira. While this is not an experiment, the results will be used and discussed in Section 6 in relation to the results of the experiments that will follow in the next subsections.

A summary of the collected Jira data is presented in Table 1. More than 12500 issues were collected over 31 projects spanning more than three years with 74 identified team members. 10889 of these issues are labeled as completed, having either the status Done or Rejected. While the number of team members is extremely high for an agile team, it is important to note that this figure includes project and delivery managers, clients, interns, and contributors from other (agile) teams. Everyone with a Jira account inside the organization (including guests) that was marked as an issue assignee, reporter, or creator is included. At the time of writing this paper the team whose data is being analyzed consists of six developers.

In total, the 15 issue statuses can be divided over 6 swim lanes (To Do, In Progress, Review, Test, Acceptance, and Done). In the To Do swim lane the issue types ‘Reserved’, ‘To Do’, ‘Backlog’, ‘Pre-Refinement’, and ‘Refinement’ can be found. The In Progress swim lane only consists of the issue type ‘In Progress’. Issue types ‘In Review’ and ‘Ready to Review’ can be found in Review, while ‘Ready to Test’, ‘In Test’, and ‘Ready to Deploy to Acceptance’ can be found in Test. Acceptance consists of the issue types ‘Ready for Acceptance’, ‘In Acceptance’, and ‘Ready to Deploy to Production’. Finally, in Done the issue types ‘Done’, and ‘Rejected’ can be found. An image of the scrum board configuration with swim lanes and their respective issue types can be found in Appendix D.

Throughout the event logs, only six issue types have been identified: Story, Epic, Task, Bug, Incident, and Sub-Task. When creating a standard issue in Jira it is not possible to immediately set the issue type as a Sub-Task. This can only be done by selecting a specific Jira issue and adding a sub-task to it. This means that a sub-task can be a child to any of the available issue types, including another sub-task. The number of times a given issue type occurs in the database is shown in Figure 22. A comparison between bug issues created and sprint issue count is presented in Figure 23.

| Database collection property | Value |
|---|---|
| Number of projects | 31 |
| Number active projects | 8 |
| Number of issues | 12654 |
| Average number of issues per project | 409 |
| Lowest issue count in project | 6 |
| Highest issues count in project | 3587 |
| Number issue types | 6 |
| Number unique issue statuses | 15 |
| Number of events | 42631 |
| Number of team members | 94 |
| Number of team member types* | 11 |
| Number completed issues | 10998 |
| Number uncompleted issues | 1656 |
| Number uncompleted issues in active projects | 1347 |
| Number uncompleted issues in nonactive projects | 309 |
| Average issue work ratio of completed issues | 59.55 |
| Average original time estimate if provided (hours) | 6.42 |
| Average time spent (completed issue) in seconds | 12879 |

Table 1: Summary of imported Jira data. *The number of team member types is included here but only collected in Section 5.3

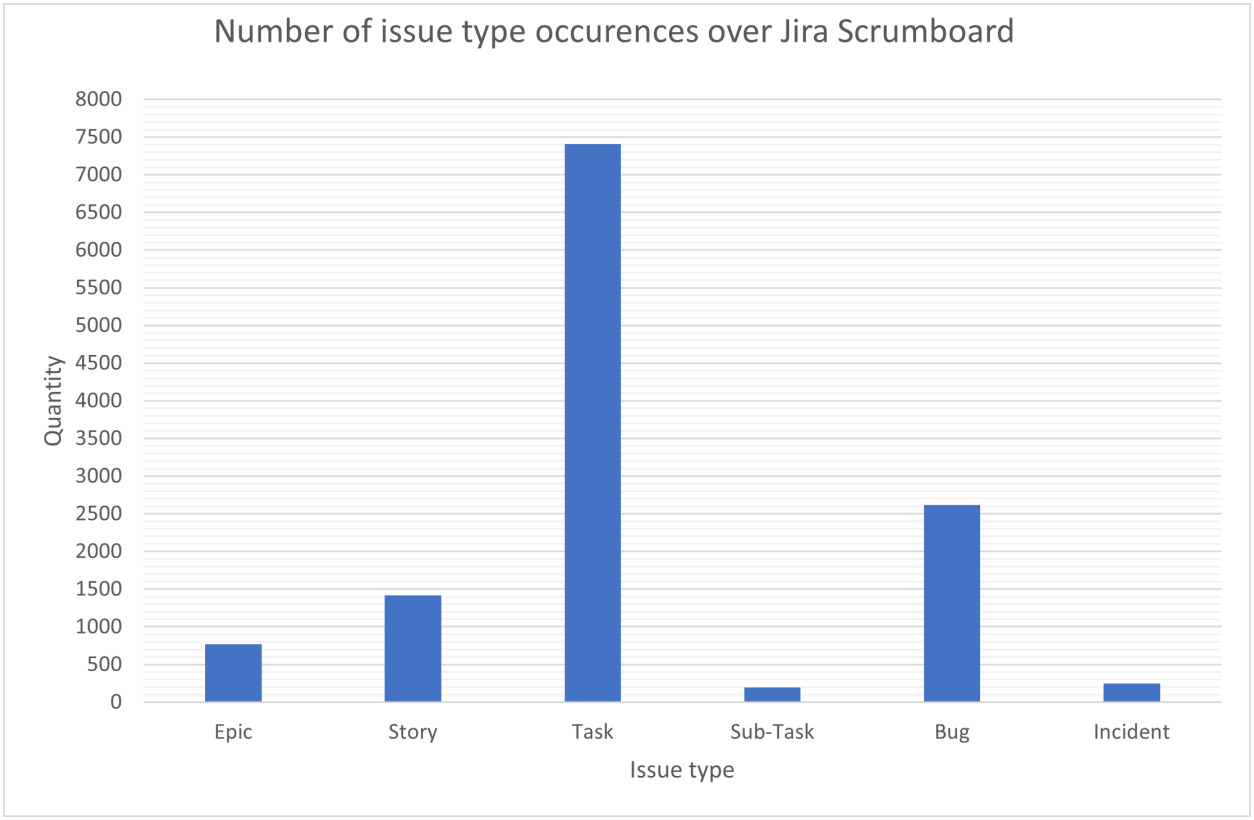



Figure 22: Number of times a given issue type occurs in the database

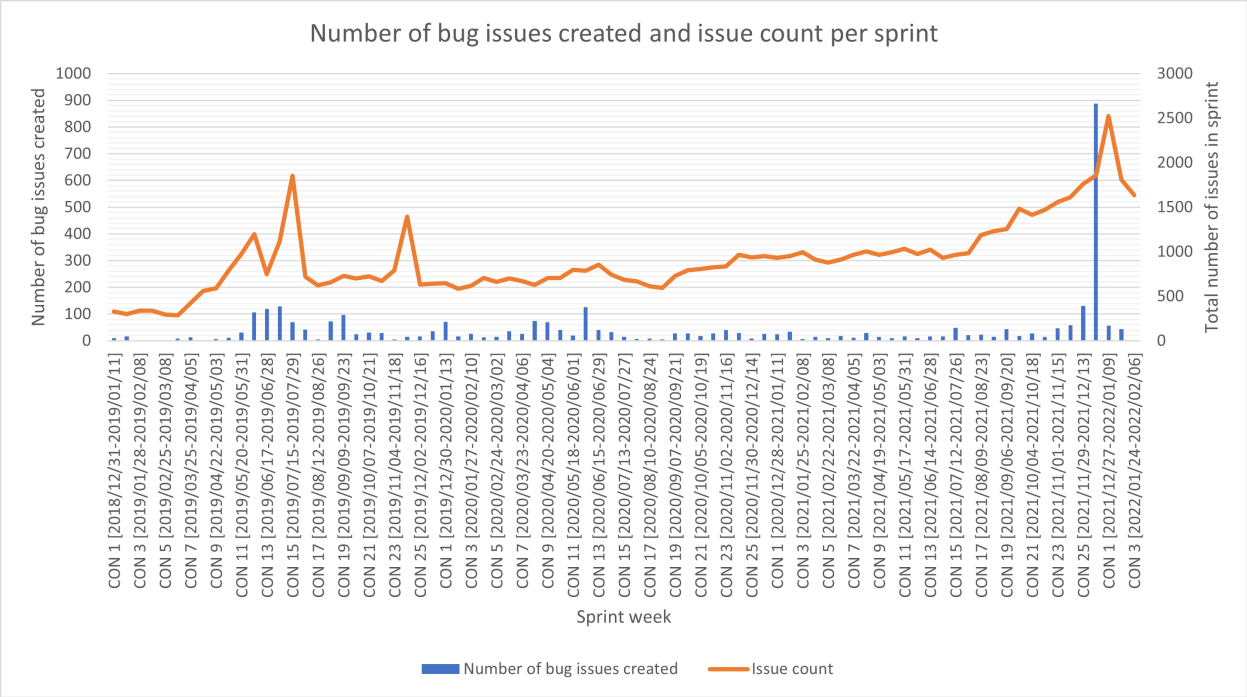



Figure 23: Number of bug issues created per sprint and sprint issue count

Active projects are defined as those present on the Scrum board at the time of the data import. In total there are 8 active projects, as of writing, with 1347 uncompleted issues. Issues are defined as completed if they are present in the Done swim lane of the Jira Scrum board (see Appendix D), so either marked as Done or Rejected. Interestingly, there are as many as 309 issues that have not been completed but are part of inactive issues.

After having imported all of the issue data, the sprint data could be imported and extended with values like issue count and the sum of the time estimate over all issues in that sprint. In total data could be collected over 81 sprints, where each sprint is two weeks long. This is a little over 3 years of data. The issue count and sum of the time estimate of these issues is presented in Figure 24.

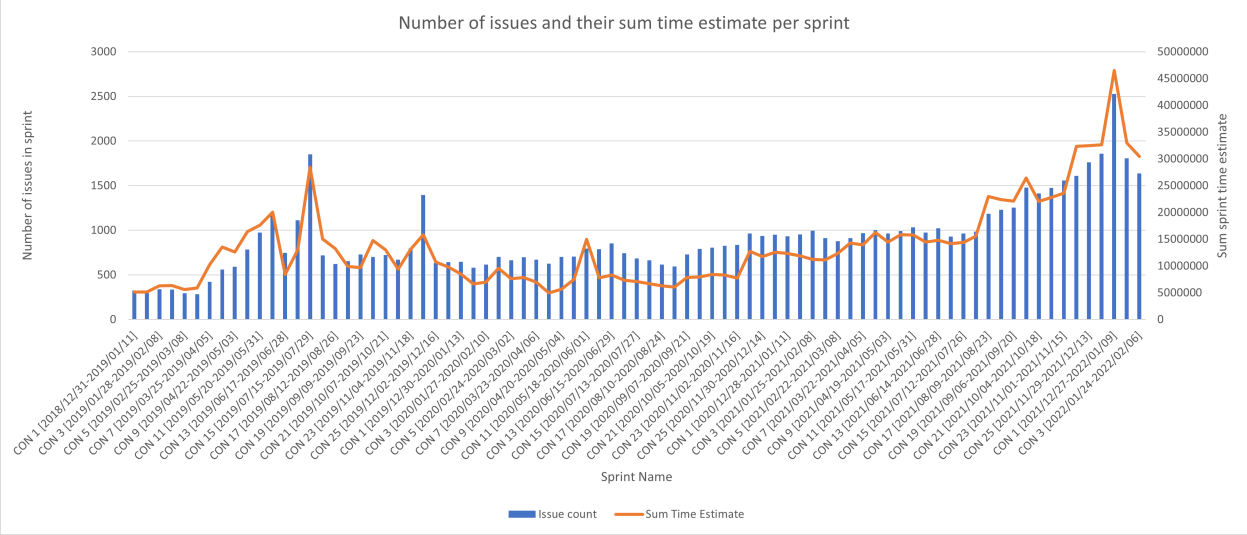



Figure 24: Issue count and sum of time estimate per sprint

During (control-flow) process discovery, how issues move across the scrum board and between the issue types will be analyzed.

## 5.2 Process discovery and conformance checking

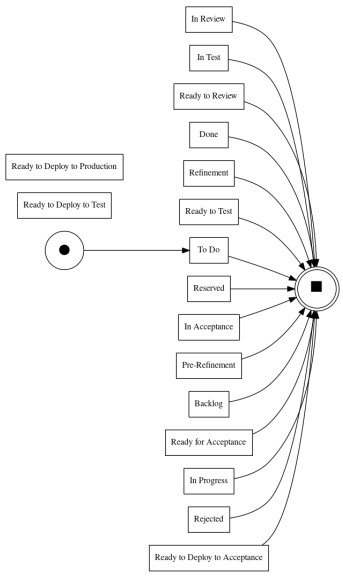
This experimentation subsection will consist of two parts: comparing the Alpha, Alpha+, Heuristics, and Inductive miners; and Conformance checking of the actual event log over the mined desired workflow model.

### 5.2.1 Comparing different discovery miners

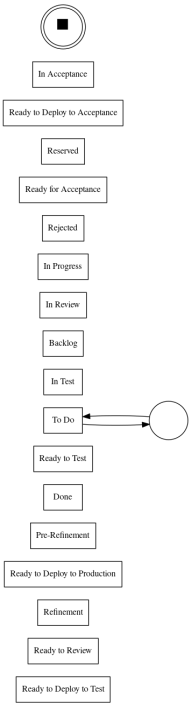
This subsection contains the first real experiment of this research paper. In this subsection, the Alpha, Alpha+, Heuristics (with varying thresholds), and Inductive (with varying thresholds) miners will be compared over the four quality metrics proposed by Rozinat et al.(Rozinat et al., 2007): fitness, precision, generalization, and simplicity - all of which are measured from 0 (low) to 1 (high).

When comparing the miners, the average quality score will be used. Simplicity is calculated over only the model itself by looking at the number of nodes and edges. This can give a slightly thwarted understanding of how well the model reflects a high-quality model if simplicity is taken into account when comparing the average quality score. To express this understandably, look at Figure 25a. This model has perfect simplicity with a value of 1, but the model tells us very little, only that issues are created, eventually move to 'To Do' and after that, they are marked completed. All other events can lead to completion, but the only causal changes we can infer from the model is that an issue is created, usually moves to 'To Do' and after something happens such that the issue closes. The Alpha+ miner (see Figure 25b) tells us even less, despite also having a perfect simplicity score of 1. For this reason, all further comparisons between miners will be done using the average score over the accuracy, precision, and generalization as these scores are defined by fitting the event log to the model.

The Alpha miner has an average quality score (ignoring simplicity) of 0.754 and the Alpha+ miner of 0.333.




(a) Model created from event log using Alpha miner




(b) Model created from event log using Alpha miner

Figure 25: Alpha and Alpha+ created models of event log

For the Inductive miner, the best average score (ignoring simplicity) came from having a very low noise threshold, between 0.0 and 0.05, with an average score of about 0.898. For a threshold of 0.0, the created models is presented in Figure 26, with a full paged size version available in Appendix **??**. While Figure 26 is difficult to read, its overall structure gives a good idea of the resulting model. This miner shows us our first silent activities (discussed in Section 2 and represented by black-boxes on the model). These are activities that are identified by the miner but do not occur in the event log. In this way, we can notice from the inductive miner that an undefined activity sets whether an issue can move from 'In Progress' (in the middle) to a completed state or first has to move through another activity. A comparison of the inductive miner scores for varying noise threshold values is presented in Figure 27.

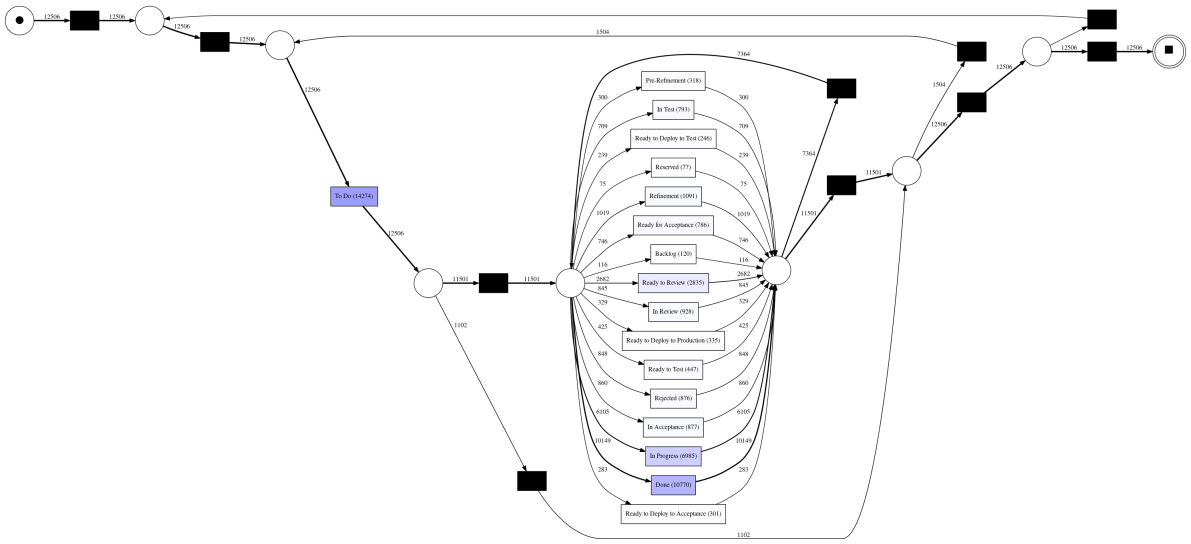



Figure 26: Model created from event log using Inductive miner with a noise threshold of 0.0. A full paged size version is available in Appendix **??**. While the model is very difficult to read, the general structure gives a good impression of the result.


(a) Quality scores of inductive miner for varying noise threshold values

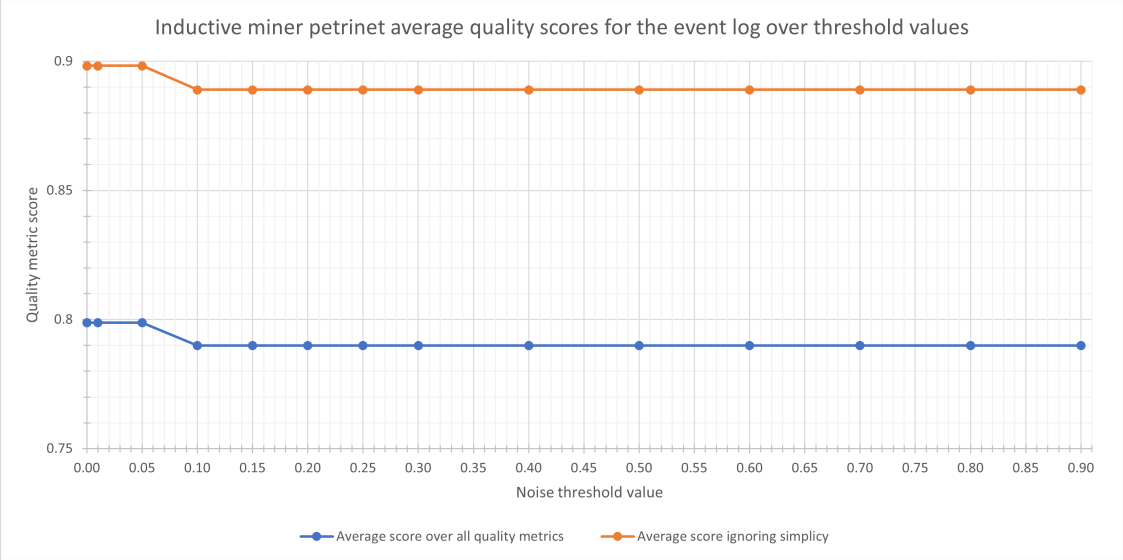

(b) Average quality score, including and ignoring simplicity, for the inductive miner for varying threshold values

Figure 27: Quality scoring for the inductive miner

Finally, there is the heuristics miner, which has to be compared over varying values for its three threshold parameters. Weijters and Ribeiro stated that these three threshold parameters are usually set to the same value(A. Weijters & Ribeiro, 2011). For this reason, setting the threshold parameters to the same value will be our initial experiment for the Heuristics miner. The best threshold parameter value was 0.99, giving an average quality score (ignoring simplicity) of 0.878. This miner is presented in Figure 29, with a full-page sized version to be found in Appendix E. While the model is very difficult to read, the general structure gives a good impression of the result. A heuristics miner with all threshold values equal to 1.0 creates a model similar to that of the Alpha miner model (see Appendix E. The comparison of quality scores for the varying threshold values is

presented in Figure 28.


(a) Quality scores of heuristics miner for varying threshold values, where all parameters are equal




(b) Average quality score, including and ignoring simplicity, for the heuristic miner for varying threshold values, where all parameters are equal

Figure 28: Quality scoring for the Heuristics miner

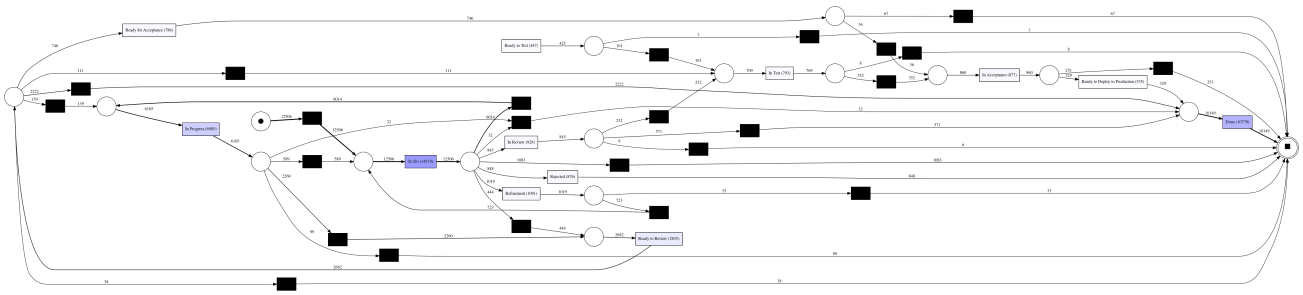
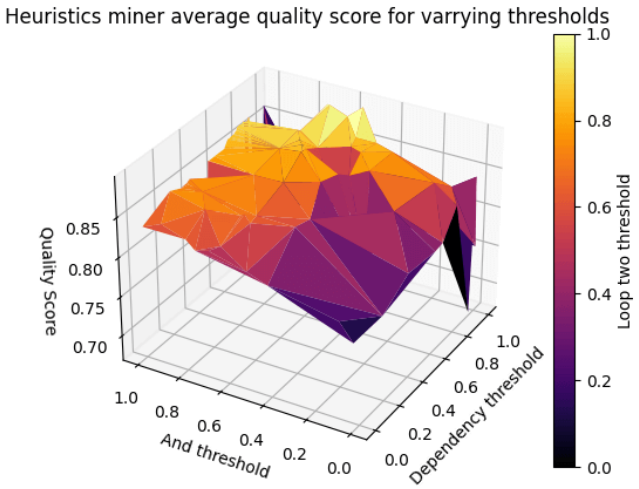
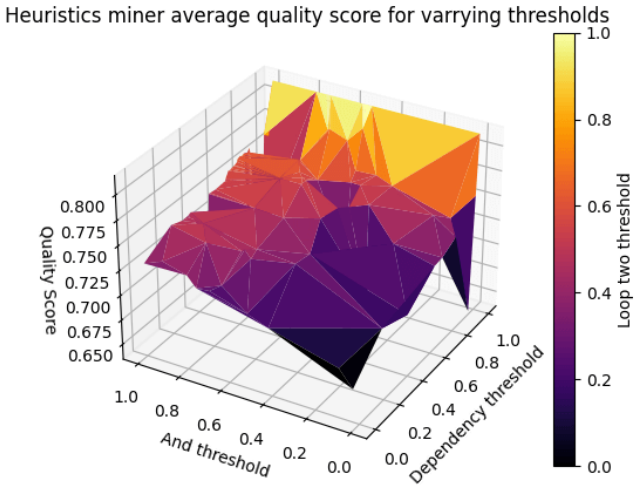


Figure 29: Model created from event log using Heuristics miner with all parameters equal at 0.99. A full paged size version is available in Appendix **??**. While the model is very difficult to read, the general structure gives a good impression of the result.

Since varying threshold values can have a large influence on the quality scores of the Heuristics miner(Suhendar et al., 2018), the three threshold values can be fine-tuned using Bayesian optimization over a Gaussian process. This was done over 100 iterations the results of which are presented in Figure 30. The Heuristics miner with the highest average quality (ignoring simplicity) that with dependency-threshold of 0.979, and-threshold of 0.535, and loop-two-threshold of 0.754, with a score of 0.896. Using mutual information regression, we can notice that the dependency-threshold has a dependency to the target variable (average score) of 0.815, the and-threshold a dependency value of 0.289, and the loop-two-threshold a dependency of 0.001 on the target variable 'average score ignoring simplicity'. A Petri net of this heuristics miner is presented in Figure 31. This model is impossible to read, but that is not the goal. Its overall structure gives a good idea of what the resulting model looks like.




(a) Surface map of the average quality score, including simplicity, over differing values of the dependency-, and- and two-loop threshold values for the heuristics miner




(b) Surface map of the average quality score, ignoring simplicity, over differing values of the dependency-, and- and two-loop threshold values for the heuristics miner

Figure 30: Quality scoring for the inductive miner

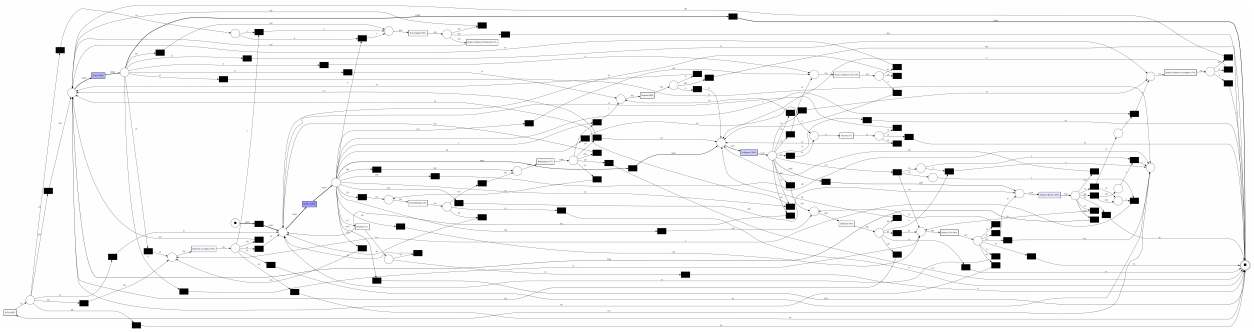


Figure 31: Model created from event log using Heuristics miner with varying threshold parameters: dependency threshold: 0.979; and threshold: 0.535; loop-two threshold: 0.754. A full paged size version is available in Appendix E. This model is impossible to read, but that is not the goal. Its overall structure gives a good idea of what the resulting model looks like, reiterating the importance of the suggested usability metric.

A complete comparison of the (average) quality scores over the different miners is presented in Figure 32.

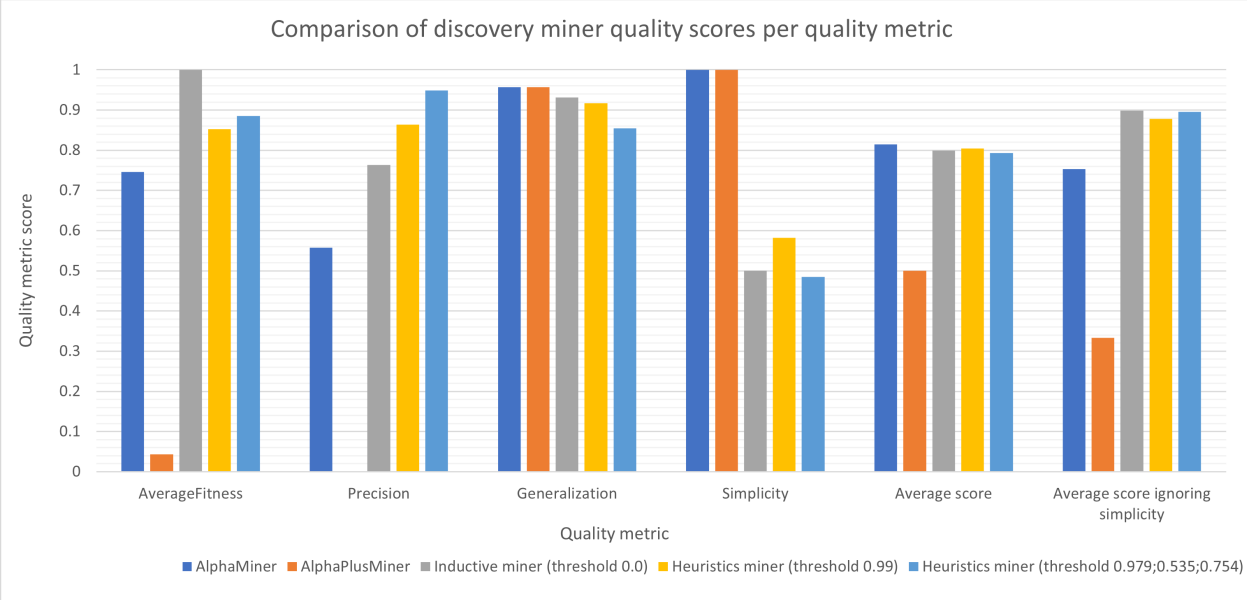



Figure 32: Comparison of (average) quality scores over the four different miners

### 5.2.2 Conformance checking with desired workflow

In this next subsection, the best performing discovery miner, the Inductive miner with a threshold value of 0.0 (see Section 5.2.1) is used to generate the desired workflow model. This is presented in Figure 33, with a full-page version available in Appendix F along with the questionnaire used to collect the data to be converted to an event log and an excerpt of the responses. The quality scoring of the mined desired workflow model, when compared to the event log, is presented in Figure 34. The average score (ignoring simplicity) was 0.858 with an average fitness per issue of 03.984.

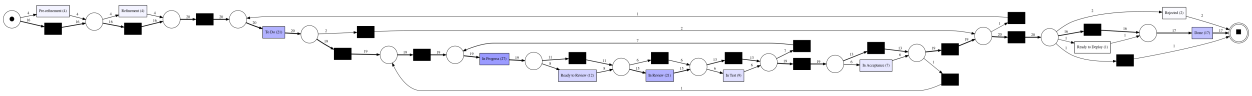


Figure 33: Inductive miner model of the desired workflow. This model again is also difficult to read, but the general structure of the model points towards the linear nature of the desired workflow.

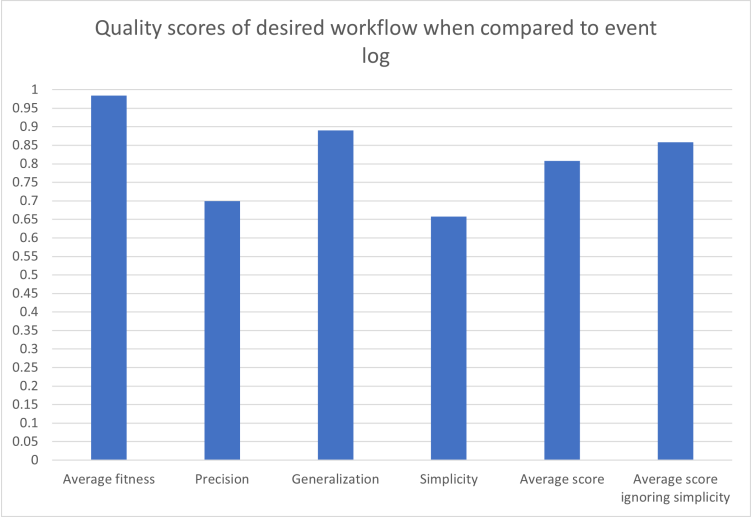



Figure 34: Quality scores of desired workflow when compared to the event log

## 5.3 Model enhancement

In this subsection, the model extension stage will be discussed. This entails extending the issues model with a fitness score as compared with the desired workflow (discovered and presented in Section 5.2.2), assigning a team member type to each team member, and finally preparing data sets for the machine learning techniques used in the next subsection. Only done issues will be used and a detailed description of the data set is provided in Section 3.2.4. While this is not an experiment, the results will be used in the next subsection and discussed in Section 6.

The quantity of team member types is presented in Figure 35.

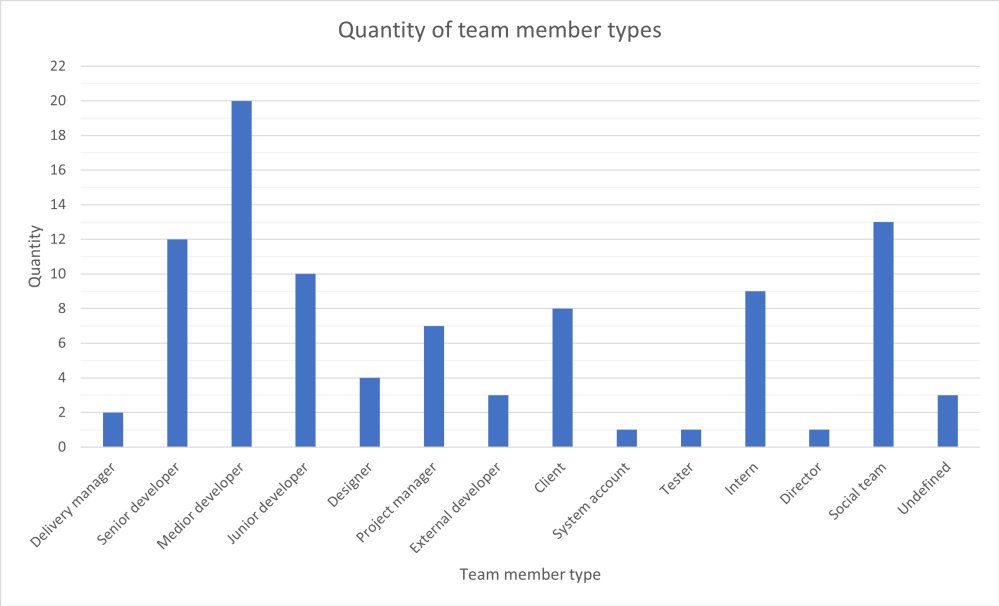



Figure 35: Quality of team member types that have contributed to Jira Scrum board

## 5.4 Operational support - applying predictive techniques

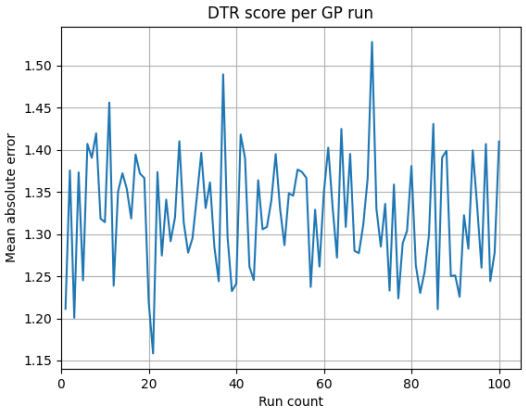
In this section, the training and test results are presented for each of the three machine learning techniques over the three prediction target variables of completed issues. The three machine learning techniques that are applied are decision tree regressor (DTR), linear support vector regressor (SVR), and multilayered perceptron regressor (MLPR). The target variables are issue work ratio, issue fitness as compared to the desired workflow, and the next issue status.

Each of the experiments starts by showing the mean absolute error per Gaussian process run while trying to optimize the hyperparameters of the respective machine learning algorithm to maximize its learning ability. With the selected best hyperparameters the algorithms are trained again over 80% of the data and tested with the remaining 20%. The algorithms are compared with each other over their $R^2$-score when predicting the test set.

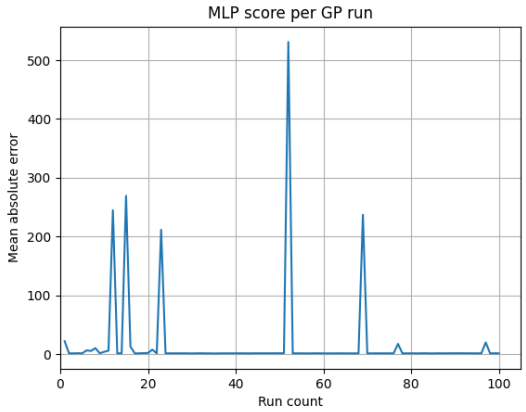
For each of the trained three machine learning techniques, the five most influential variables are presented along with their importance. For MLPR this is difficult as a neural network is notorious for being a black box. To get around this issue the mutual information score will be used.

### 5.4.1 Predicting work ratio

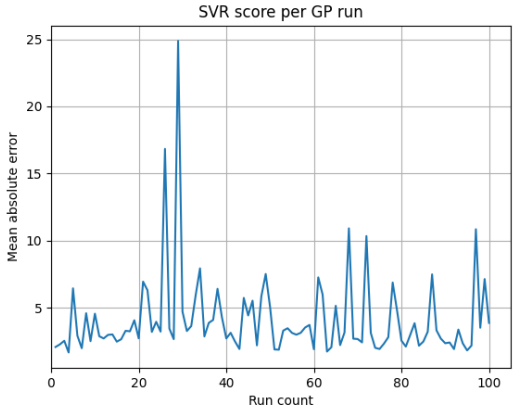
Figure 36 shows the mean absolute error per GP run while trying to find the optimum algorithm hyperparameters when predicting work ratio.



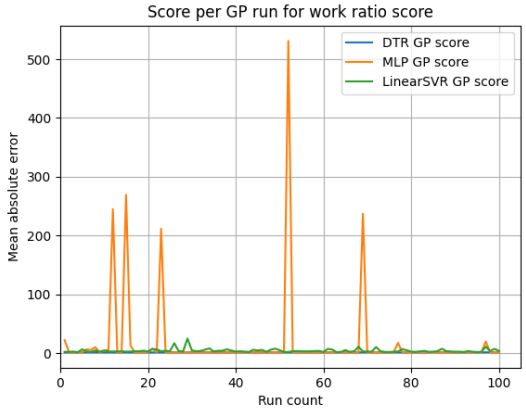

(a) Mean absolute error per run count of GP for DTR




(b) Mean absolute error per run count of GP for MLP




(c) Mean absolute error per run count of GP for SVR




(d) Mean absolute error per run count of GP for DTR, MLPR and SVR

Figure 36: Mean absolute error of machine learning technique over run count when optimizing hyperparameters using Gaussian process (GP) when estimating work ratio

| **ML algorithm** | $R^2 - score$ |
|---|---|
| DTR | -0.0000127 |
| MLPR | 0.0029654 |
| SVR | -24.0202063 |

Table 2: $R^2$-score of machine learning techniques when estimating work ratio

| Algorithm | Ranking metric | Rank | Variable | Value |
|---|---|---|---|---|
| DTR | GINI | 1 | N/A | 0 |
| | | 2 | N/A | 0 |
| | | 3 | N/A | 0 |
| MLPR | Mutual information | 1 | Project Issue Number | 1.1348999 |
| | | 2 | Time Original Estimate | 0.9261337 |
| | | 3 | Size Description | 0.2954805 |
| SVR | Weight coefficient | 1 | Time Spent | 0.0093151 |
| | | 2 | Time Estimate | 0.0003280 |
| | | 3 | Time Original Estimate | -0.0002978 |

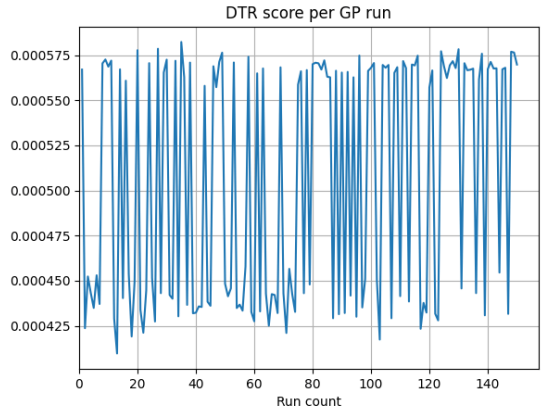
Table 3: Feature importance when predicting work ratio as defined by the DTR, SVR or mutual information score. N/A shows that the algorithms has identified no importance features and as such all features have an importance value of 0

| Feature | Low value | Low value target value | High value | High value target value |
|---|---|---|---|---|
| Project issue number | 1 | 70 | 4000 | 70 |
| Time original estimate | 0 | 70 | 28800 (8hours) | 70 |
| Size description | 1 | 70 | 100 | 70 |

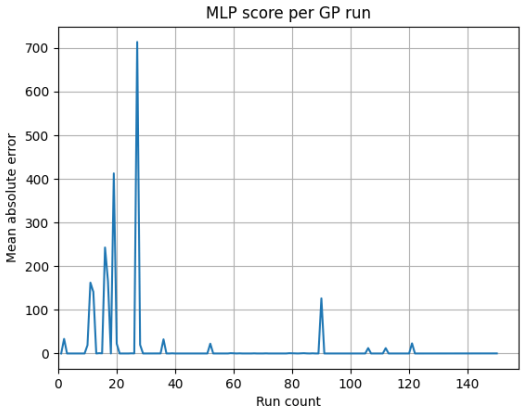
Table 4: Results of estimated work ratio when for varying feature values while keeping all other features constant. As we can see with our given MLPR model, changing values of the features with the highest mutual information does not alter estimations; reflecting an inaccurate model.

### 5.4.2 Predicting issue fitness

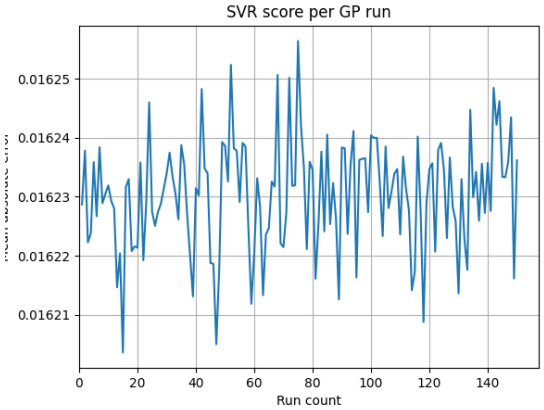
Figure 36 shows the mean absolute error per GP run while trying to find the optimum algorithm hyperparameters when predicting fitness.



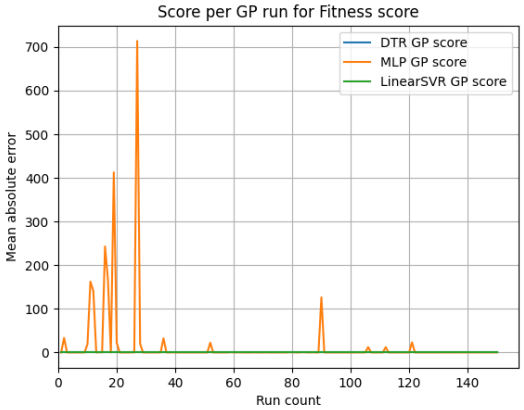

(a) Mean absolute error per run count of GP for DTR




(b) Mean absolute error per run count of GP for MLP




(c) Mean absolute error per run count of GP for SVR




(d) Mean absolute error per run count of GP for DTR, MLPR and SVR

Figure 37: Mean absolute error of machine learning technique over run count when optimizing hyperparameters using Gaussian process (GP) when estimating fitness

| **ML algorithm** | $R^2 - score$ |
|---|---|
| DTR | -0.3909058 |
| MLPR | -0.5335134 |
| SVR | -538.9894137 |

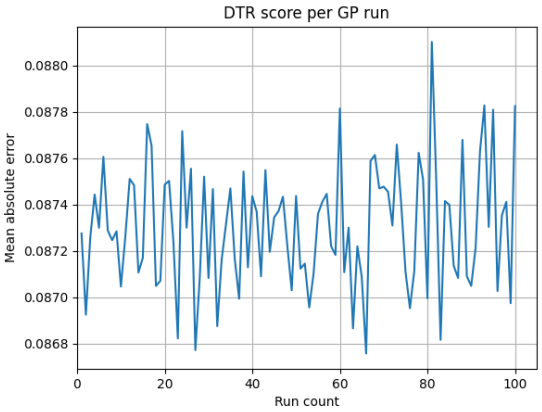
Table 5: $R^2$-score of machine learning techniques when estimating fitness

| Algorithm | Ranking metric | Rank | Variable | Value |
|---|---|---|---|---|
| DTR | GINI | 1 | N/A | 0 |
| | | 2 | N/A | 0 |
| | | 3 | N/A | 0 |
| MLPR | Mutual information | 1 | Project Issue Number | 1.2382365 |
| | | 2 | Size Description | 0.3723560 |
| | | 3 | Sprint Sum Estimated Time | 0.3371259 |
| SVR | Weight coefficient | 1 | N/A | 0 |
| | | 2 | N/A | 0 |
| | | 3 | N/A | 0 |

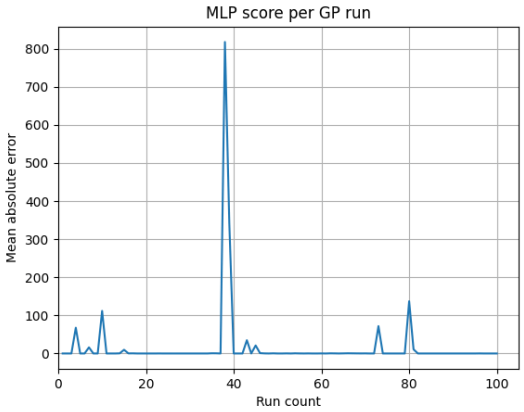
Table 6: Feature importance when predicting fitness as defined by the DTR, SVR or mutual information score. N/A shows that the algorithms has identified no importance features and as such all features have an importance value of 0

#### 5.4.3 Predicting next issue state (status)

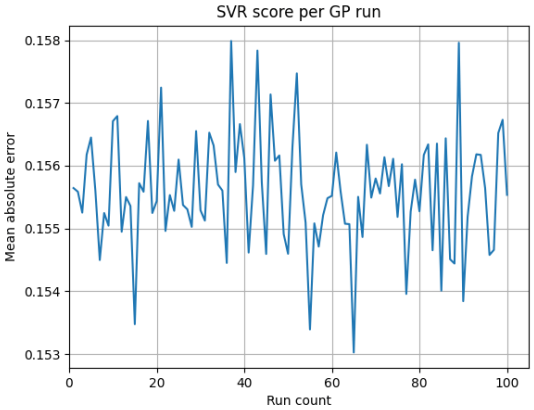
Figure 36 shows the mean absolute error per GP run while trying to find the optimum algorithm hyperparameters when predicting the next issue state.



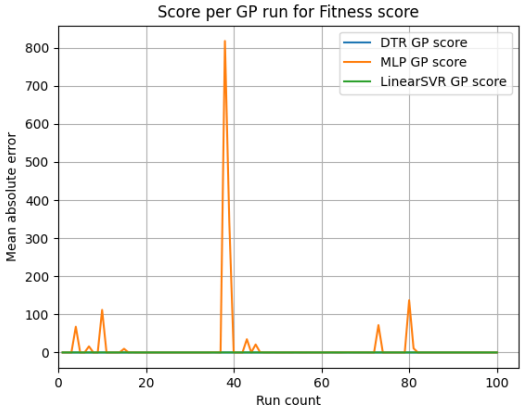

(a) Mean absolute error per run count of GP for DTR




(b) Mean absolute error per run count of GP for MLP




(c) Mean absolute error per run count of GP for SVR




(d) Mean absolute error per run count of GP for DTR, MLPR and SVR

Figure 38: Mean absolute error of machine learning technique over run count when optimizing hyperparameters using Gaussian process (GP) when estimating next state of issue

| **ML algorithm** | $R^2 - score$ |
|---|---|
| DTR | -0.0000874 |
| MLPR | 0.0377237 |
| SVR | -3.0018992 |

Table 7: $R^2$-score of machine learning techniques when estimating next state

| Algorithm | Ranking metric | Rank | Variable | Value |
|---|---|---|---|---|
| DTR | GINI | 1 | N/A | 0 |
| | | 2 | N/A | 0 |
| | | 3 | N/A | 0 |
| MLPR | Mutual information | 1 | Current Status | 0.4501339 |
| | | 2 | Time Since To Do | 0.2988391 |
| | | 3 | Sprint Issue Count | 0.1792191 |
| SVR | Weight coefficient | 1 | N/A | 0 |
| | | 2 | N/A | 0 |
| | | 3 | N/A | 0 |

Table 8: Feature importance when predicting next issue status as defined by the DTR, SVR or mutual information score. N/A shows that the algorithms has identified no importance features and as such all features have an importance value of 0

| Feature | Low value | Low value target value | High value | High value target value |
|---|---|---|---|---|
| Current status | 3 (To do) | 10 (Ready for Acceptance) | 13 (In Acceptance) | 10 (Ready for Acceptance) |
| Time since to do | 1800 (0.5 hours) | 9 (In Test) | 259200 (3 weeks) | 9 (In Test) |
| Sprint issue count | 1 | 9 (In Test) | 2500 | 9 (In Test) |

Table 9: Results of estimated next issue state when for varying feature values while keeping all other features constant. As we can see with our given MLPR model, changing values of the features with the highest mutual information does not alter estimations; reflecting an inaccurate model.

# 6 Discussion

This Section will present a discussion of the results of experimentation and how it relates to the literature. This will consist of interpreting the data, suggesting the implications, discussing the limitations, and providing recommendations.

The discussion will be approached along the lines of the $L^*$ life-cycle model to keep with the structure used in Sections 12, 4 and 5. This will allow the experiments and their results to be discussed structured to their respective subsections. After having discussed the results the research question will be readdressed and answered.

## 6.1 Identifying areas of improvement through exploration of the Jira process data

During the $L^*$ stage of extracting and exploring knowledge in Section 5.1, Jira data was imported and a summary of that data was presented. From this data, several note-worthy points about the data and the workings of the team need to be addressed.

### 6.1.1 Enforcing workflows to avoid outliers and simplify workflow

Based on initial exploratory analysis, there are 15 possible issue statuses allowed over the investigated team's Scrum board. While some issue statuses occur more often than others, each of these statuses are used by at least one issue in the issue history. Possible movement between these statuses is not defined or enforced within Jira, despite Jira offering this service. A given issue can therefore theoretically move from 'Done' back to 'To Do', despite a 'Done' issue being completed and not adjustable again. By allowing this type of undesirable movement, the team opens itself up to uncontroversial or undesired workflows, without directed team discussion of the issue. A simple recommendation can be to enforce desired workflows, allowing issues that required undesired status changes to be flagged by the team as a whole, forcing it to be discussed during a stand-up. Erdem and Demirörs (Erdem et al., 2018) also emphasized that formalizing and standardizing development processes is essential in reducing inconsistencies, instability, and unpredictability of the development process.

### 6.1.2 High bug count leading to over-planned sprints

The next important issue to note is the high number of Bug issue types in the database (see Figure 22). About 20.7% of all issues are bugs. This is undoubtedly an undesirably large number of bugs compared to the total number of issues being worked on. A deep dive into why such a high number of bugs occurs can be recommended. Reducing bug quantity can greatly improve team performance while also allowing the team to continue moving forward. If completed projects still lead to identified bugs by the service desk (internal team that processes client requests after delivery), the investigated team will continually be pulled back into completed projects, without actually having time for them.

It can also be noted that there has been a steady increase in the number of issues, and their associated sum of time estimates, over the last year. The last year has been fairly interesting for

the investigated team as the team continued to expand in a slightly turbulent fashion. While team members joined, several also left. This can also be a result of the work-from-home policies due to COVID-19 (and variants). The number of issues peaked just before the new year of 2022. This is not too strange as projects are often aimed to be completed before the new year. The relationship between the number of issue bugs created in a sprint and a rise in the number of issues in the next sprint is shown in Figure 23, where a high number of bugs created leads to a high number of issues in the next few sprints. This could be reminiscent of turbulence in the team and could be an interesting area to investigate; the effect of increased bug count of project planning for the next few weeks. This could also reflect bugs moving from one sprint to the next, or bugs pushing planned issues from one sprint to the next, piling on the workload. Either way, a clear area of improvement would be to reduce bug count. A simple way to reduce bug count would be to apply test-driven development methods. While unit tests have been integrated over the last year to improve CI/CD, full integration tests have not. Full integration tests would help identify problems in the review and test phases of issues. This further emphasizes the importance of tests. This is built upon in Section 6.2.

It should also be noted that there is a large number of uncompleted issues in completed projects. These issues should be either completed or moved to rejected as needed and required by the agile process. After a project is completed the backlog should be clear of issues.

### 6.1.3 Non-agile number of contributors: how desirable is this?

It can also be noted that a large number of different team members contribute, by being either an assignee, creator, or reporter of an issue, to the issues on the investigated teams Scrum board. In an agile workspace, teams are generally very small, up to 8 members, allowing for a targeted approach to issues. Having a lot of different members contribute to the issues on the Scrum board can reduce issue understanding before development or tackling of that issue. From Figure 35 it can be seen that up to 10 different people from management (including executive level) have contributed to the Scrum board. As many as 13 members from an entirely different team (the social team) have also contributed. The high number of developers that have contributed (up to 42) is also worrisome if the team is supposed to be agile.

This does not immediately mean that the team is not working agile, but rather that the Scrum board might not be directed at the team level, as can usually be assumed, but rather at (inter-team) project or portfolio level. It can not be assumed that this is undesirable and does not reflect agile work as it does reflect the agile principles of work being directed at people. With active iterations of an MVP, different teams and member types (including customers or the social team) can reflect and add desired functionality directly into the teams' backlog. The important part that has to be decided by the investigated team is whether these elements added from outside the team are of high quality and reflect what needs to be done. This could also explain the high number of uncompleted issues in completed projects as the (non-)functional requirements defined by those (externally added) issues have already been handled and thus point to work overlap. The contributions of external project members like clients of the social team might also effect the suitability of feature or issue descriptions, again emphasizing the importance of checking input quality.

### 6.1.4 Defining (non-)functional descriptions in agile fashion to allow clear project overview and avoid misunderstandings

Finally, many issue descriptions do not follow the proposed agile structure of defining tasks or stories. These structures should follow the lines of $< usertype >$ wanting to $< performaction >$ to achieve a $< result >$ for stories, or for a feature $< action >< leads\ to >< object >$. Issue summaries generally simply describe the problem or a state a desired component, like "Cluster algorithm" (a feature which can easily be better described or even split into several issues).

When working on projects, you will often notice that you are the one creating your own issues in Jira and therefore generally have a good understanding of what is meant or required, leading to colloquial or lackadaisical issue descriptions. This is problematic for two reasons. Firstly, you will not be the only one working on a project at any one time and have to be able to efficiently discuss your work during standups. Clear and complete issue descriptions will allow for concise and effective task overview for project managers during standups. Secondly, requiring clear feature descriptions has to be considered while taking into account the a high number of contributors and their roles. Without clear issue definition structuring, contributors are likely to misunderstand requirements. Clear feature descriptions assist in understanding the relevant done criteria. Erdem and Demirörs (Erdem & Demirörs, 2017) identified done citeria as often not being upheld, leading to miss matches between desired workflow and actual workflow.

## 6.2 Comparison of control-flow discovery miners: why the heuristics miner is the most appropriate to use

During the process discovery and conformance checking phase of experimentation (see Section 5.2.1), different process miners are used to created models of the event log and are compared using the quality metrics of fitness, precision, generalization, and simplicity. The control-flow discovery miners that were used are the Alpha, Alpha+, Heuristics, and Inductive miners.

### 6.2.1 Usability: the missing miner quality metric

It is quickly noticed that a high simplicity value does not necessarily add value or understanding. This can be noticed by the resulting models of the Alpha and Alpha+ miners (see Figure 25) which have very high simplicity but relatively low accuracy and precision. In fact, for the heuristics, miner accuracy and precision drastically decrease as simplicity increases (see Figure 28). Unfortunately, as simplicity scores decrease, so does the readability of the model as model complexity greatly increases. This would suggest a missing quality metric of *usability*, which would be hard to define due to its subjectivity, but would add extra depth to model comparison.

The highest scoring miner, based on model quality ignoring simplicity, is the inductive miner. Despite not being in the list of the most popular miners (see Figure 6), it does appear to be the best functioning miner for the event log. Unfortunately, the models created by the inductive miner do not give great insight into the unique moments of issues through the Scrum board. This is done better by the Heuristics miner models (see Figures 29 and 31). This again points towards the need for a new miner quality metric measuring usability.

### 6.2.2 Why the Alpha miners perform poorly while Inductive and Heuristics miners perform well

The low average quality scores of both the Alpha and Alpha+ miners can be explained by the inherent limitations these sibling algorithms have. Both cannot handle loops, are unable to identify silent transitions, cannot guarantee soundness and are weak to noise. The heuristics and inductive miners on the other hand, can identify silent transitions and short loops while the inductive miner can also guarantee soundness.

The need for models to be able to handle loops is self-evident in that issues can move backwards in the workflow, for example from In Review back to In Progress. Further more, the identification of silent transitions is required as not all issues will flow the standard sequence through all possible states as some states can and likely will be skipped (not all issues require going through the Refinement state for example). Finally, there is the issue of noise in the data. Some issues in the data follow drastically unexpected workflows (jumping from Done back to In Progress for example), a problem that might arise from an enforced workflow management in Jira. These domain based requirements suggest an explain to the fitness and precision weaknesses of the Alpha and Alpha+ miners.

Soundness of a petrinet looks at the guarantee of proper completion of firing sequences in the model. This should not be a problem since no Jira issue can be in two states at a single time due to the sequential processing nature of software feature issues across the Scrum board. This can explain the almost negligible average quality metric values between the optimized inductive and heuristics miners since both miners can handle noise, loops, and identify silent transactions. The difference between these miners then comes in the suggested new quality metric of usability. While the heuritics miner has a lower simplicity, with much low readability, the produced model does show a more sequential nature to the processing of issues. This might explain why, according to the literature, the heuristics miner the the more popular choice control-flow miner as its usability is suggested to be higher. Dos Santos Garcia et al. also suggested the heuristics miner to be the most popularly used miner due to the algorithms robustness and capability to "deal with noise and exceptions in unstructured processes"(dos Santos Garcia et al., 2019).

This higher usability can then be used to better analyze processes and sequential steps along with frequency count. From the Heuristics miners, we can notice that relatively few issues move into either the 'In Review' or 'In Acceptance' phases, 845 and 709 of 12506 respectively. This can explain the high bug rate at intervals over all the sprints and further emphasizes the importance of test-driven development with unit tests *and* (full) integration testing combined with active review sessions. While this might not necessarily reflect missing integration tests or hastily performed review, it could point towards limited feature description refinement, which later can lead to mislabeled 'bugs'.

### 6.2.3 Event log data fitting well into the desired workflow

When comparing the event log with the desired workflow, it is clear that most issues fit in well with the desired workflow. But average fitness, as well as generalization, are high, hitting 0.98 and 0.88 respectfully. Precision is relatively low, on the other hand, suggesting under-fitting of the

data. The desired workflow model is very clear though and identifies silent activities (see Figure 33). While not all silent activities are interesting, take the one corresponding to 'Ready to Review' as an example, there are activities that hold weight, like those representing circular movement through the model (the silent activity after 'In Acceptance' for example). These desired possible workflow moments can be enforced in Jira but have not yet been done by the team. Enforcing the possible workflows can potentially help increase precision while also reducing movement error and identifying problematic issues immediately by way of blocked issue placement.

The investigated team having a fairly simple (desired) workflow fits well into the agile goal of effectiveness. The issues analyzed were from a single scrum board and not a collection of issues flowing across multiple boards at an inter-team or portfolio level. This inherently makes the (desired) flows simplistic due to a single scrum board needing to be orderly and manageable. A simple workflow is desirable as discussed by Caldeira and Cardoso (Caldeira et al., 2019) since complex process models often reflect a lack of effectiveness. If the dataset were to be extended with additional data to reflect inter-team issues or portfolio-level data, the process models will inherrently become more complex as they have to reflect the workflows of all the teams being managed. This will likely also greatly reduce the usability of any mined control-flow model.

Having compared the different control-flow process mining algorithms across the quality metrics, either the inductive or heuristics miners can be suggested to use when analyzing the Scrum board workflow of an agile software development team. If we take the newly suggested quality metric of usability into account, the heuristics miner would be the best choice due to the created model giving better insight into the actual workflow of the team.

## 6.3 Understanding why the applied machine learning techniques failed to accurately predict the target variables and potential improvements

In this section, the results from the predictive machine learning techniques and why they were unsuccessful are discussed. Using a SVR, DTR, and MLPR over the data set (defined in Section 3.2.4) an attempt was made to predict the work ratio, fitness, and next status of issues moving through the Scrum board. Unfortunately, these predictive techniques were unsuccessful in accurately predicting any of the target variables.

There are many reasons why machine learning models fail to converge or produce desired accuracy scores. Ways to improve accuracy scores include, but are not limited to, data preprocessing, feature engineering, feature selection, algorithm selection, hyperparameter tuning and simply using more data. While an attempt was made to apply most of these improvement methods, they will be discussed individually, reiterating application and suggesting potential improvements for further research.

### 6.3.1 Handling outliers and missing values

Data preprocessing often involves handling missing values and identifying outliers in an attempt to reduce model bias. The simple way in which this was applied was to only include completed tasks in the dataset. Incomplete tasks would, for example, not include values like eventual work ratio,

nor would be the fitness of the issue be accurate.

While an attempt was made at avoiding missing values, outliers were not identified and removed. This is a potential further improvement. Furthermore, the dataset consists independent variables extracted directly from Jira where they all have values defined. These values were not extended with additional data (outside of Jira) or transformed (both of which are discussed in the following subsections).

### 6.3.2 Feature engineering

Feature engineering often takes on the form of feature transformation, where values are normalized or grouped, and feature creation, where new variables are defined based on other variables or summarized. Aside from adjusting the *timeSpent* value of *Rejected* issues (which was kicked up to a maxium values), no attempt was made at feature transformation.

For feature creation, several computed variables were included and are summarized in the *InterIssueInformation* table in Figure 13. These are computed fields that summarize differences in sprint information.

Feature engineering is a potential way to improve accuracy during further research by summarizing variables and discovering hidden relationships.

### 6.3.3 Feature selection

During feature selection, statistical methods are used, like PCA or Mutual Information, to reduce dimensionality and identify relationships and correlations between variables. While the use of Mutual Information scores is touched upon in Section 5.4 to identify potential features to teak when providing recommendations, feature selection was not applied in this paper.

### 6.3.4 Selecting the right ML algorithm

Many different machine learning algorithms exist and selecting the right one to use can greatly increase model accuracy. The diversity of machine learning techniques to use in big data analysis of agile software development is summarized well by Biesialska et al.(Biesialska et al., 2021) who identified more than 12 possible algorithms to used divided over 4 general types of problems.

Selecting the right algorithm to use is tough. The algorithms that were selected to be used were based on the literature, identifying three algorithms that showed promising results in analyzing agile software development team data: SVR, MLPR and DTR. A further exploration of possible algorithms to apply could be interesting for further research.

### 6.3.5 Hyperparameter tuning

Almost all machine learning algorithms are accompanied by different parameters that set and define the finer workings of that algorithm. Discovering optimal values for those parameters can greatly increase accuracy. To this end Baysian Optimization was used to tune the parameters of the different machine learning algorithms. The paper on which this approach is based dates back almost a decade. In recent years there have been other suggested algorithms and the exploration of these could help

improve learning and accuracy. This would not likely greatly improve accuracy though. Baysian Optimization has proven effective(Jasper Snoek & Adams, 2012) and the accuracy of the applied machine learning techniques were so bad that a slightly different selection of hyperparameters would not have greatly improved accuracy.

### 6.3.6 More data, always more data

Finally, another common way to improve model accuracy is by adding more data to use to train and test. This can take on two dimensions: increasing sample size or, closely related to feature engineering, increase sample dimensionality.

The dataset consisted of almost 27,000 items all of which were relatively homogeneous. Due the the size of the dataset and the similarity of the items, increasing the sample size would probably not have greatly improved model accuracy.

Instead, the low accuracy scores of the machine learning models could point towards the necessary variables, or the necessary combination of variables, not being present in the dataset. Different variables from different sources could help enhance the dataset and result in better accuracy scores.

Software development is a vary dynamic field of work with new technologies and methodologies constantly being introduced. Furthermore, running a team over the span of several years bring an inherent sense of volatility, with team members joining and leaving, or the projects (and their types) changing. Despite Jira being a powerful tool, it cannot store and process all of these changing factors, even less so if logging and usage is not enforced.

Having said this a potential source of extra data within Jira could be the worklogs in which team members log the actual time they work on issues. This will obviously better reflect time spent on issues instead of using the less reliable measure of logging when tasks change status. But this again requires tool usage enforcement.

## 6.4 Stimulating innovation by way of data-driven DevOps technique comparison

The inherent rigidity of companies often stems from the fact that people themselves are often rigid. As soon as someone has a certain 'correct' method in their minds, it is often difficult to convince them to try new things. This is especially the case when financial and time risk mitigation are central to most, if not all, business decisions. Despite this, we have come to know and understand that calculated risk taking is essential in driving innovation. Furthermore, continual innovation remains quintessential in business stability and growth.

Selecting which new paradigm to embrace and which dogma to shun is extremely difficult and requires careful consideration. A nudge in the right direction can be provided by using data-driven reasoning to select which (hybrid) development methodology to encompass. Olszewskatelal et al. suggested using project management tool data to measure the effect of agile transformations on responsiveness, throughput, workflow distribution and quality(Olszewska et al., 2016). By using these metrics the effectiveness of seemingly intangible or abstract changes could actively be measured and compared in a quantifiable manner. This provides a data-driven way of comparing methods

with each other.

Marques et al. suggested using process mining to measure implementations and processes of agile teams(Marques et al., 2018), providing a clear and concise way to gather the data to quantify the magnitude of change. In this paper an architectural design and case study implementation on using process mining to analyse a teams workflow data was developed and discussed. While this application aimed (and failed) to accurately apply predictive techniques to provide recommendations, the application was able to give (visual) insight into the workflows as well as provide quantifiable data around responsiveness (time estimates), throughput (work ratio) and quality (number of bugs).

The architectural design (and its implementation) in itself suggests a clear way of collecting, parsing and gaining knowledge from the Jira Scrum board data. This data can then be used, as suggested by Marques et al. and Olszewskatelal et al., to measure and compare different (agile) methodologies and the transformation between them. Properly applying this tool should make it easier to break the rigid thinking of teams or the organization itself by providing clear data to measure changes in methodology. This will make methodological changes more palpable and thus hopefully stimulate teams to continually try new things, moving in the direction that measures the most success for that team.

### 6.5 Answering the research question

The research question for this paper is: ***How can process mining be used to suggest Scrum improvements for an agile software development team, and what kind of architectural design is required to investigate and provide these recommendations?***

Process mining provides a data-driven way to gain visual insights into actual processes of teams and organizations. This form of data analysis can be used to uncover hidden processes and lead to better business intelligence. Business intelligence and analytics are generally divided into three fields: descriptive, predictive, and prescriptive. Olszewska et al. identified a fourth: adaptive analytics. These fields can be used to gain deeper insight into businesses while estimating future events or values and providing recommendations to improve value by, for example, reducing time- or cost-overrun. An example field where time- and cost-overrun are notoriously high is that of software development. To be able to make accurate predictions, data needs to be collected and analyzed which is where process mining comes into play. Using process mining, further insight can be gained while providing extra information with which to make predictions. These predictions can then be used to make recommendations to avoid undesirable workflows and the potentially resulting time- or cost-overrun. These recommendations can in turn help improve DevOps by identifying and improving areas of overrun.

To help guide any process mining project, the undisputed God Father of process mining van der Aalst suggested using the $L^*$-lifecycle(Van Der Aalst, 2011). This would divide the project up into five stages, extending and enhancing the three types of process mining. In keeping with this project structure, this paper has suggested an architectural design for research projects using Jira backlog data. This design is presented in Section 4.

*How do process mining techniques and standards compare?* For the event log generated by the Jira data of the .NET team analyzed for this paper, the inductive miner performed best across the

quality metrics. The heuristics miner, with optimized values, came in a close second. As discussed above the quality metrics used for analyzing models developed by discovery miners do seem to lack a metric for model usefulness.

*How is process mining applied in (agile) software development?* Two general application areas of process mining in agile software development have been identified: process mining of user actions to help software development teams improve the system or product being developed; and process mining of the actions software development team members take to analyze, identify and improve the processes team members go through when developing software.

*How is process mining used in predictive and prescriptive analytics?* Process mining can be used in predictive and prescriptive analytics to answer different c-level questions, like instance duration, delay identification, and recommended execution path. These predictive methods include "decision trees, case based-reasoning, recommender systems, and neural networks"(dos Santos Garcia et al., 2019). These techniques can be applied to be able to provide real-time decision support, allowing for better business agility. Extending data models and applying machine learning techniques allow researchers and teams to detect and predict unwanted workflows and provide recommendations. Depending on the machine learning technique applied, this method, combined with statistics, can be used to identify feature importance and effect on predictions, providing a prescriptive-like tool to help direct teams as they process unwanted workflows and try to find alternatives.

*What machine learning techniques are industry standards in analyzing process mining data and other retrieved process data?*Process mining is still a very young field of research, dating not much more than a few decades. Due to the unmatured nature of this research field, no set collection of machine learning techniques has been defined. Different machine learning algorithms to use include Bayesian indicators, decision trees, support vector machines, and neural networks. During process enhancement, for the developed data set, three machine learning techniques are used to attempt to predict issue work ratio, fitness, and next issue status. These algorithms are decision tree regressors, support vector regressors, and multilayer perceptron regressors (neural network). All three machine learning techniques used failed to predict issue work ratio, fitness, or the next issue status.

# 7 Conclusion

This section will serve to conclude this research paper. To this end, the paper will be summarized and potential future work will be discussed.

The objective of this paper was to provide an architectural design for process mining of agile software development teams Scrum board workflow data. To help guide this research objective, the following research question was defined: How can process mining be used to suggest Scrum improvements for an agile software development team, and what kind of architectural design is required to investigate and provide these recommendations?

This was achieved by providing a proof of concept through a case-study investigation of an actual software development team that uses Jira. During the period of this case study I worked as a part-time software engineer within the team being investigated. Using the Jira Api to extract team data since the usage of Jira was introduced over three years before, data over 31 projects, spanning 12,650+ issues, was collected and a machine learning data set consisting of 26800+ elements was built.

The contribution of this paper is threefold. First, an architectural design, based on the literature, for process mining Jira Scrum board data and providing recommendations is developed. Second, this architectural design is implemented in the form of a prototype process mining tool. Finally, by comparing and contrasting different process mining algorithms, a recommendation is made as to which process miner should be used.

To conclude, Jira data can be extremely unstructured reflecting the human nature of the process of software development. Alignment of actual human workflow to the workflow in Jira - self improvement, where the structure inherently comes more from the social aspect of tool usage. This means that usage of this tool requires team wide structuring of processes.

## 7.1 Future research

Unfortunately, the predictive machine learning techniques were unable to accurately predict the proposed target variables. A possible area of future work would be a deeper dive into collectible data, extending the machine learning data set, to be able to make accurate predictions of target variables. Using these trained machine learning techniques combined with statistics, feature importance can be defined and how these features affect the target variable, providing prescriptive recommendations. While this was attempted, unfortunately, the data set was not large enough or the required features for accurate prediction were not present.

An area for data set improvement would be to not only use the event log but the actual work log, containing logged hours of work by the team members. These work logs and cases can be clustered using time warping (an example of this was discussed in Section 2). While this can be calculated from the issue change logs, coming it with an organizational perspective of process mining can allow the identification of bottlenecks in the process. If reviews could only be performed by senior developers, for example, these work logs combined with an organizational perspective can highlight the bottlenecks in ‘Review’ while also showing the effectiveness of these seniors outside of reviews.

Another potential area of further research would be to use the architectural design and its developed solution to measure agile transformations within teams. The application could be used to provide quantitative data that allows teams to compare different methodologies and find the one that best suits them in a data-driven fashion.

Finally, by using work logs combined with an organizational, control-flow, and resource perspectives, a development simulation environment could be created. With a simulation environment further AI techniques, including reinforcement learning can be applied to suggest work time coordination for team members. Think of if a team member should drop the issue they are working on at the moment to fix a bug, or if more value is contributed to the team by first finishing up what is being worked on before moving on to the bug. Creating this simulation environment with which to test proposed actions provides a great recommendation system.

## References


Agrawal, R., Gunopulos, D., & Leymann, F. (1998). Mining process models from workflow logs. In *International conference on extending database technology* (pp. 467–483).

Anttiroiko, A.-V., Valkama, P., & Bailey, S. J. (2014). Smart cities in the new service economy: building platforms for smart services. *AI & society*, *29*(3), 323–334.

Anwer, F., Aftab, S., Shah, S. M., & Waheed, U. (2017). Comparative analysis of two popular agile process models: extreme programming and scrum. *International Journal of Computer Science and Telecommunications*, *8*(2), 1–7.

Auth, G., JokischPavel, O., & Dürk, C. (2019). Revisiting automated project management in the digital age–a survey of ai approaches. *Online Journal of Applied Knowledge Management (OJAKM)*, *7*(1), 27–39.

Baier, T., Mendling, J., & Weske, M. (2014). Bridging abstraction layers in process mining. *Information Systems*, *46*, 123–139.

Bassil, Y. (2012). A simulation model for the waterfall software development life cycle. *arXiv preprint arXiv:1205.6904*.

Belhajjame, K., Corcho, O., Garijo, D., Zhao, J., Missier, P., Newman, D. R., ... others (2012). Workflow-centric research objects: A first class citizen in the scholarly discourse. In *Sepublica@ eswc* (pp. 1–12).

Bergenthum, R., Desel, J., Lorenz, R., & Mauser, S. (2007). Process mining based on regions of languages. In *International conference on business process management* (pp. 375–383).

Biermann, A. W., & Feldman, J. A. (1972). On the synthesis of finite-state machines from samples of their behavior. *IEEE transactions on Computers*, *100*(6), 592–597.

Biesialska, K., Franch, X., & Muntés-Mulero, V. (2021). Big data analytics in agile software development: A systematic mapping study. *Information and Software Technology*, *132*, 106448.

Boehm, B. W. (1976). Software engineering education: Some industry needs. *Software Engineering Education*, 13–19. doi: 10.1007/978-1-4612-9898-4_4

Boon, G. C., & Stettina, C. J. (2022). A case for data-driven agile transformations: Can longitudinal backlog data help guide organizational improvement journeys? In *International conference on agile software development* (pp. 114–130).

Bose, R. J. C., Mans, R. S., & van der Aalst, W. M. (2013). Wanna improve process mining results? In *2013 ieee symposium on computational intelligence and data mining (cidm)* (pp. 127–134).

Caldeira, J., e Abreu, F. B., Reis, J., & Cardoso, J. (2019). Assessing software development teams' efficiency using process mining. In *2019 international conference on process mining (icpm)* (pp. 65–72).

Chui, M., Manyika, J., & Miremadi, M. (2018). What ai can and can't do (yet) for your business. *McKinsey Quarterly*, *1*, 97–108.

Cloudscene. (2018, Apr). *Data centers in amsterdam - where are they and who operates them?* Retrieved from https://cloudscene.com/news/2017/12/data-centers-in-amsterdam/

Davis, A. M., Bersoff, E. H., & Comer, E. R. (1988). A strategy for comparing alternative software development life cycle models. *IEEE Transactions on Software Engineering*, *14*(10), 1453–1461.

De Medeiros, A. A., & van der Aalst, W. M. (2008). Process mining towards semantics. In *Advances in web semantics i* (pp. 35–80). Springer.

De Smedt, J., De Weerdt, J., & Vanthienen, J. (2014). Multi-paradigm process mining: Retrieving better models by combining rules and sequences. In *Otm confederated international confer-*

*ences" on the move to meaningful internet systems"* (pp. 446–453).

Despa, M. L. (2014). Comparative study on software development methodologies. *Database systems journal*, *5*(3), 37–56.

De Weerdt, J., De Backer, M., Vanthienen, J., & Baesens, B. (2012). A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs. *Information Systems*, *37*(7), 654–676.

De Weerdt, J., Schupp, A., Vanderloock, A., & Baesens, B. (2013). Process mining for the multi-faceted analysis of business processes—a case study in a financial services organization. *Computers in Industry*, *64*(1), 57–67.

dos Santos Garcia, C., Meincheim, A., Junior, E. R. F., Dallagassa, M. R., Sato, D. M. V., Carvalho, D. R., ... Scalabrin, E. E. (2019). Process mining techniques and applications–a systematic mapping study. *Expert Systems with Applications*, *133*, 260–295.

Erdem, S., & Demirörs, O. (2017). An exploratory study on usage of process mining in agile software development. In *International conference on software process improvement and capability determination* (pp. 187–196).

Erdem, S., Demirörs, O., & Rabhi, F. (2018). Systematic mapping study on process mining in agile software development. In *International conference on software process improvement and capability determination* (pp. 289–299).

Evermann, J., Rehse, J.-R., & Fettke, P. (2017). Predicting process behaviour using deep learning. *Decision Support Systems*, *100*, 129–140.

for Applied Information Technology, F. I. (n.d.). *Handling event data.* Retrieved from https://pm4py.fit.fraunhofer.de/documentation

Gill, S. S., Tuli, S., Xu, M., Singh, I., Singh, K. V., Lindsay, D., ... others (2019). Transformative effects of iot, blockchain and artificial intelligence on cloud computing: Evolution, vision, trends and open challenges. *Internet of Things*, *8*, 100118.

Group, T. O. (n.d.). *Archimate 3.1 specification.* Retrieved from https://pubs.opengroup.org/architecture/archimate3-doc/

Group, X. W., et al. (2016). Ieee standard for extensible event stream (xes) for achieving interoperability in event logs and event streams. *IEEE Std*, *1849*, 1–50.

Günther, C. W., & Van Der Aalst, W. M. (2007). Fuzzy mining–adaptive process simplification based on multi-perspective metrics. In *International conference on business process management* (pp. 328–343).

Gupta, M., Sureka, A., & Padmanabhuni, S. (2014). Process mining multiple repositories for software defect resolution from control and organizational perspective. In *Proceedings of the 11th working conference on mining software repositories* (pp. 122–131).

Hagerty, J. (2017). planning guide for data and analytics. *Gartner Inc*, 13.

Homayounfar, P. (2012). Process mining challenges in hospital information systems. In *2012 federated conference on computer science and information systems (fedcsis)* (p. 1135-1140).

Jansen-Vullers, M. H., van der Aalst, W. M., & Rosemann, M. (2006). Mining configurable enterprise information systems. *Data & Knowledge Engineering*, *56*(3), 195–244.

Jasper Snoek, H. L., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *NIPS'12: Proceedings of the 25th International Conference on Neural Information Processing Systems*, *2*, 2951—2959.

Jones, D. (2001). A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimizations*, *21*, 345-383.

Josey, A., Lankhorst, M., Band, I., Jonkers, H., & Quartel, D. (2016). An introduction to the archimate® 3.0 specification. *White Paper from The Open Group*.

Kaplan, R. S., & Norton, D. (2021, Nov). *The balanced scorecard-measures that drive performance.* Retrieved from https://hbr.org/1992/01/the-balanced-scorecard-measures-that-drive-performance-2

Koch, T., & Windsperger, J. (2017). Seeing through the network: Competitive advantage in the digital economy. *Journal of Organization Design*, *6*(1), 6.

Kraskov, A., Stögbauer, H., & Grassberger, P. (2011). Erratum: estimating mutual information [phys. rev. e 69, 066138 (2004)]. *Physical Review E*, *83*(1), 019903.

Lakshmanan, G. T., Shamsi, D., Doganata, Y. N., Unuvar, M., & Khalaf, R. (2015). A markov prediction model for data-driven semi-structured business processes. *Knowledge and Information Systems*, *42*(1), 97–126.

Lamma, E., Mello, P., Montali, M., Riguzzi, F., & Storari, S. (2007). Inducing declarative logic-based models from labeled traces. In *International conference on business process management* (pp. 344–359).

Larman, C. (2004). *Agile and iterative development: a manager's guide.* Addison-Wesley Professional.

Lau, H. C., Ho, G. T., Zhao, Y., & Chung, N. (2009). Development of a process mining system for supporting knowledge discovery in a supply chain network. *International Journal of Production Economics*, *122*(1), 176–187.

Leau, Y. B., Loo, W. K., Tham, W. Y., & Tan, S. F. (2012). Software development life cycle agile vs traditional approaches. In *International conference on information and network technology* (Vol. 37, pp. 162–167).

Lee, C., Choy, K. L., Ho, G. T., & Lam, C. H. (2016). A slippery genetic algorithm-based process mining system for achieving better quality assurance in the garment industry. *Expert systems with applications*, *46*, 236–248.

Leemans, S. J., Fahland, D., & van der Aalst, W. M. (2013). Discovering block-structured process models from event logs-a constructive approach. In *International conference on applications and theory of petri nets and concurrency* (pp. 311–329).

Lepenioti, K., Bousdekis, A., Apostolou, D., & Mentzas, G. (2020). Prescriptive analytics: Literature review and research challenges. *International Journal of Information Management*, *50*, 57–70.

Ly, L. T., Maggi, F. M., Montali, M., Rinderle-Ma, S., & van der Aalst, W. M. (2013). A framework for the systematic comparison and evaluation of compliance monitoring approaches. In *2013 17th ieee international enterprise distributed object computing conference* (pp. 7–16).

Ly, L. T., Maggi, F. M., Montali, M., Rinderle-Ma, S., & van der Aalst, W. M. (2015). Compliance monitoring in business processes: Functionalities, application, and tool-support. *Information systems*, *54*, 209–234.

Marques, R., da Silva, M. M., & Ferreira, D. R. (2018). Assessing agile software development processes with process mining: A case study. In *2018 ieee 20th conference on business informatics (cbi)* (Vol. 1, pp. 109–118).

Márquez-Chamorro, A. E., Resinas, M., Ruiz-Cortés, A., & Toro, M. (2017). Run-time prediction of business process indicators using evolutionary decision rules. *Expert Systems with Applications*, *87*, 1–14.

Mittal, M., & Sureka, A. (2014). Process mining software repositories from student projects in an undergraduate software engineering course. In *Companion proceedings of the 36th international conference on software engineering* (pp. 344–353).

Moniruzzaman, A., & Hossain, D. S. A. (2013). Comparative study on agile software development methodologies. *arXiv preprint arXiv:1307.3356*.

Monostori, L. (2003). Ai and machine learning techniques for managing complexity, changes and uncertainties in manufacturing. *Engineering applications of artificial intelligence*, *16*(4), 277–291.

Olszewska, M., Heidenberg, J., Weijola, M., Mikkonen, K., & Porres, I. (2016). Quantitatively measuring a large-scale agile transformation. *Journal of Systems and Software*, *117*, 258–273.

ONeil, C. (2016). *Weapons of math destruction: how big data increases inequality and threatens democracy*. Penguin Books.

Polato, M., Sperduti, A., Burattin, A., & de Leoni, M. (2014). Data-aware remaining time prediction of business process instances. In *2014 international joint conference on neural networks (ijcnn)* (pp. 816–823).

Poppendieck, M., & Poppendieck, T. (2003). *Lean software development: an agile toolkit*. Addison-Wesley.

Popplestone, R. (1968). The design philosophy of pop-2. *Machine Intelligence*, *3*, 393–402.

Rojas, E., Munoz-Gama, J., Sepúlveda, M., & Capurro, D. (2016). Process mining in healthcare: A literature review. *Journal of biomedical informatics*, *61*, 224–236.

Royce, W. W. (1987). Managing the development of large software systems: concepts and techniques. In *Proceedings of the 9th international conference on software engineering* (pp. 328–338).

Rozinat, A., de Medeiros, A. K. A., Günther, C. W., Weijters, A., & van der Aalst, W. M. (2007). The need for a process mining evaluation framework in research and practice. In *International conference on business process management* (pp. 84–89).

Rubin, V., Günther, C. W., Van Der Aalst, W. M., Kindler, E., Van Dongen, B. F., & Schäfer, W. (2007). Process mining framework for software processes. In *International conference on software process* (pp. 169–181).

Rubin, V., Lomazova, I., & Aalst, W. M. v. d. (2014). Agile development with software process mining. In *Proceedings of the 2014 international conference on software and system process* (pp. 70–74).

Senderovich, A., Di Francescomarino, C., & Maggi, F. M. (2019). From knowledge-driven to data-driven inter-case feature encoding in predictive process monitoring. *Information Systems*, *84*, 255–264.

Shani, A. H. M., Sarno, R., Sungkono, K. R., & Wahyuni, C. S. (2019). Time performance evaluation of agile software development. In *2019 international seminar on application for technology of information and communication (isemantic)* (pp. 202–207).

Shrivastava, P. (1995). Environmental technologies and competitive advantage. *Strategic management journal*, *16*(S1), 183–200.

*sklearn.svm.svr*. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html

Souza, G. C. (2014). Supply chain analytics. *Business Horizons*, *57*(5), 595–605.

Suhendar, A., Wisudiawan, G. A. A., & Herdiani, A. (2018). Implementation of process mining with flexible heuristics miner algorithm to support information system audit. *Sustainable Collaboration in Business, Technology, Information and Innovation (SCBTII)*, *1*(1).

Suriadi, S., Andrews, R., ter Hofstede, A. H., & Wynn, M. T. (2017). Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs. *Information systems*, *64*, 132–150.

Van den Broucke, S. K., & De Weerdt, J. (2017). Fodina: A robust and flexible heuristic process discovery technique. *decision support systems*, *100*, 109–118.

Van der Aalst, W. (2012a). Process mining. *Communications of the ACM*, *55*(8), 76–83.

Van der Aalst, W. (2012b). Process mining: Overview and opportunities. *ACM Transactions on Management Information Systems (TMIS)*, *3*(2), 1–17.

Van Der Aalst, W. (2016). Data science in action. In *Process mining* (pp. 3–23). Springer.

Van der Aalst, W., Adriansyah, A., De Medeiros, A. K. A., Arcieri, F., Baier, T., Blickle, T., ... others (2012). Process mining manifesto. In *International conference on business process management* (pp. 169–194).

Van Der Aalst, W. M. (2011). Process mining: discovering and improving spaghetti and lasagna processes. In *2011 ieee symposium on computational intelligence and data mining (cidm)* (pp. 1–7).

Van der Aalst, W. M., De Medeiros, A. A., & Weijters, A. J. (2005). Genetic process mining. In *International conference on application and theory of petri nets* (pp. 48–69).

Van der Aalst, W. M., Reijers, H. A., Weijters, A. J., van Dongen, B. F., De Medeiros, A. A., Song, M., & Verbeek, H. (2007). Business process mining: An industrial application. *Information Systems*, *32*(5), 713–732.

Van der Aalst, W. M., & Song, M. (2004). Mining social networks: Uncovering interaction patterns in business processes. In *International conference on business process management* (pp. 244–260).

Van der Aalst, W. M., Weijters, A., & Maruster, L. (2002). *Workflow mining: Which processes can be rediscovered* (Tech. Rep.). Citeseer.

Visual-Paradigm. (n.d.-a). *Full archimate viewpoints guide.* Retrieved from https://www.visual-paradigm.com/guide/archimate/full-archimate-viewpoints-guide/#product-viewpoint

Visual-Paradigm. (n.d.-b). *What is archimate.* Author. Retrieved from https://www.visual-paradigm.com/guide/archimate/what-is-archimate/

Weijters, A., & Ribeiro, J. T. S. (2011). Flexible heuristics miner (fhm). In *2011 ieee symposium on computational intelligence and data mining (cidm)* (pp. 310–317).

Weijters, A., van Der Aalst, W. M., & De Medeiros, A. A. (2006). Process mining with the heuristics miner-algorithm. *Technische Universiteit Eindhoven, Tech. Rep. WP*, *166*, 1–34.

Weijters, A. J., & Van der Aalst, W. M. (2003). Rediscovering workflow models from event-based data using little thumb. *Integrated Computer-Aided Engineering*, *10*(2), 151–162.

Wen, L., Wang, J., & Sun, J. (2006). Detecting implicit dependencies between tasks from event logs. In *Asia-pacific web conference* (pp. 591–603).

Williams, L. (2010). Agile software development methodologies and practices. In *Advances in computers* (Vol. 80, pp. 1–44). Elsevier.

Womack, J. P., Jones, D. T., & Roos, D. (2007). *The machine that changed the world: The story of lean production–toyota's secret weapon in the global car wars that is now revolutionizing world industry.* Simon and Schuster.

Yang, S., Dong, X., Sun, L., Zhou, Y., Farneth, R. A., Xiong, H., ... Marsic, I. (2017). A data-driven process recommender framework. In *Proceedings of the 23rd acm sigkdd international conference on knowledge discovery and data mining* (pp. 2111–2120).

Yu, T., & Zhu, H. (2020). *Hyper-parameter optimization: A review of algorithms and applications.*

Zayed, M. A., & Farid, A. B. (2016). The discovery of the implemented software engineering process using process mining techniques. *Int. J. Adv. Comput. Sci. Appl*, *1*(7), 279–286.

# A Agile vs Lean Principles

A Comparison of Agile and Lean Principles

| | | Lean principles | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | Eliminate waste | Build quality in | Create knowledge | Defer commitment | Deliver fast | Respect people | Optimize the whole |
| Agile principles | Early and continuous delivery | X | | X | | X | | |
| | Welcome changing requirements | | | X | X | | | X |
| | Short iterations/releases | X | | X | | X | | |
| | Business/developers work together daily | | | X | X | X | | X |
| | Motivated individuals | | | | | X | X | |
| | Face-to-face communication | | | X | | | X | |
| | Working software as measure of progress | X | | X | | | | |
| | Sustainable development | | | | | | X | |
| | Technical excellence/good design | | | X | X | | | X |
| | Simplicity | | X | | X | | | |
| | Self-organizing teams | | | | | X | X | X |
| | Reflections | | | X | | X | X | X |

Figure 39: Comparison between Agile and Lean principles (Poppendieck & Poppendieck, 2003)

# B Comparison Scrum and XP

| Features | Extreme Programming | Scrum |
|---|---|---|
| Development Approach | Iterative and incremental | Iterative and incremental |
| Project Size | Small | All |
| Team Size | 2 to 10 | Multiple teams of less than 10 members |
| Team Activities | Yes; Planning game, Pair programming, Collective code ownership etc. | No |
| Iteration/Sprint Duration | 1 to 3 weeks | 4 weeks |
| Stakeholder's Involvement | Throughout the process | Not defined |
| Communication Style | Oral, through standup meetings | Oral, through Scrum meeting |
| Project Management | No | Yes; Practices for project management are available |
| Physical Environment | Co-located teams | Not defined |
| Abstraction Mechanism | Object oriented | Object oriented |
| Focus | Towards engineering aspects | Towards management and productivity aspects |
| Response to Change | Quick | Quick |
| Requirement Elicitation | User stories and on-site customer practices are used | Not defined |
| Distinction Among Different Requirements (Functional, Non-functional) | Not defined | Not defined |
| Documentation | Less | Less |
| Upfront design Document | No | Not defined |
| Design Flexibility | Start from Simple design that can be changed using refactoring. | Focus on simple design |
| Development order defined by | Customer | Scrum Team |
| Development Style | Adaptive | Adaptive |
| Code Ownership | Whole team | Not defined |
| Changes During Iteration | Allowed | Not allowed |
| Acceptance Criteria | Defined | Defined |
| Feedback | Span from minutes to months | Span over a month |
| Testing | Unit testing, integration testing, acceptance testing | Not defined |
| Structured Review meetings | No | No |
| Validation Technique | Functional Testing and Acceptance Testing | Not defined |
| Quality Assurance Activities | Test first approach | Not defined |
| Coding Standards | Properly defined | Not defined |
| Software Configuration Practices | Not defined | Not defined |
| Support for Distributed Projects | No | Not defined |
| Process Management | No | No |

Figure 40: Comparison between Scrum and XP (Anwer et al., 2017)

# C Entity relationship diagram of Jira data

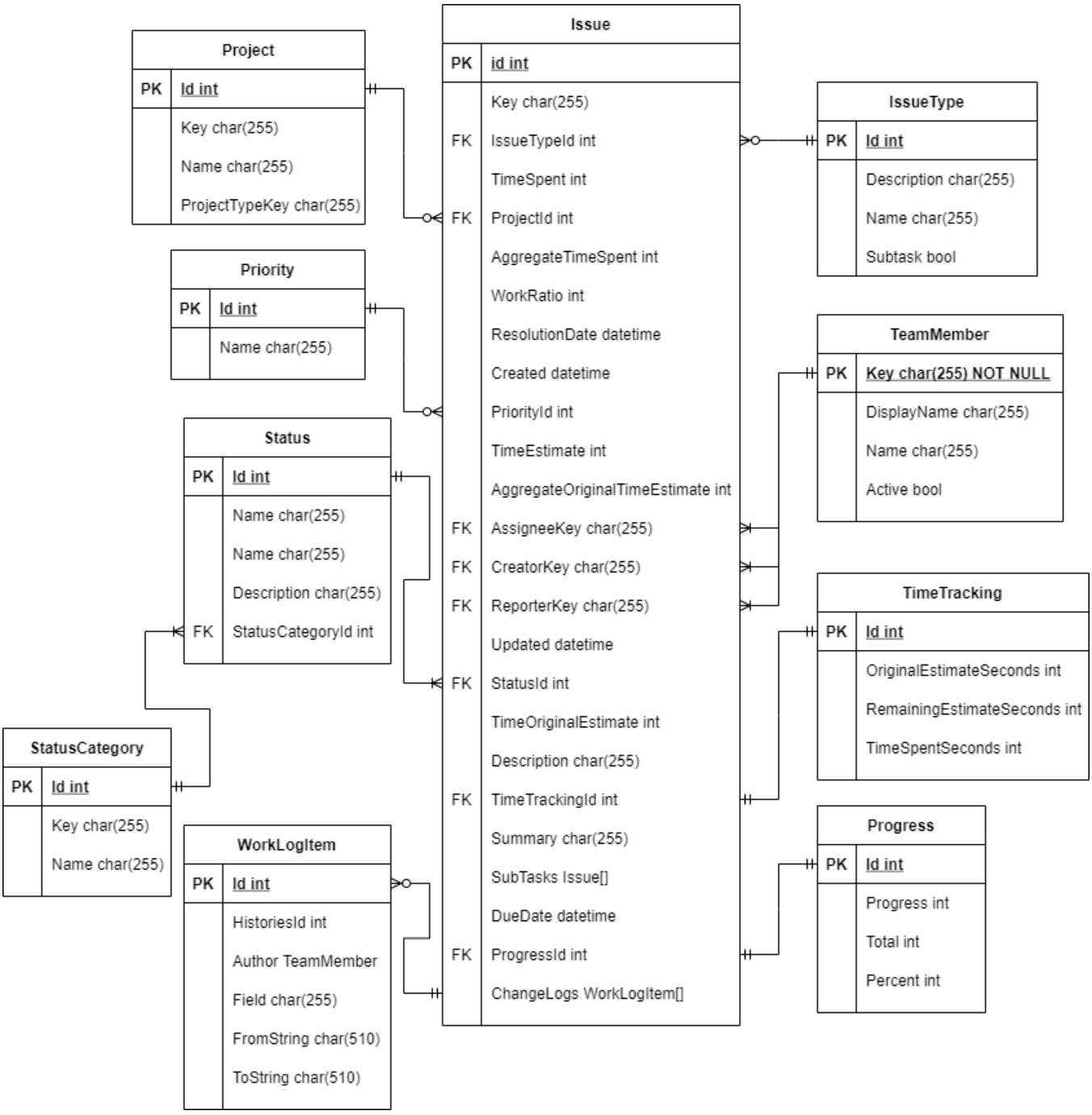



Figure 41: UML ERD showing the relationship between entities retrieved from Jira

# D Scrum board layout and issue creation

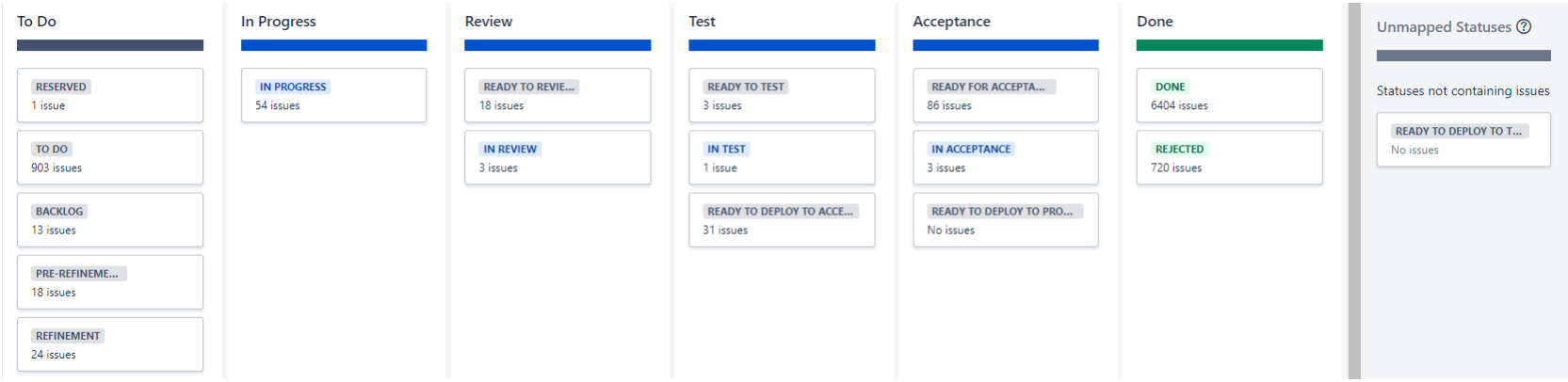



Figure 42: Scrum board layout of the agile team whose data is being analyzed, including swim lanes and issue types

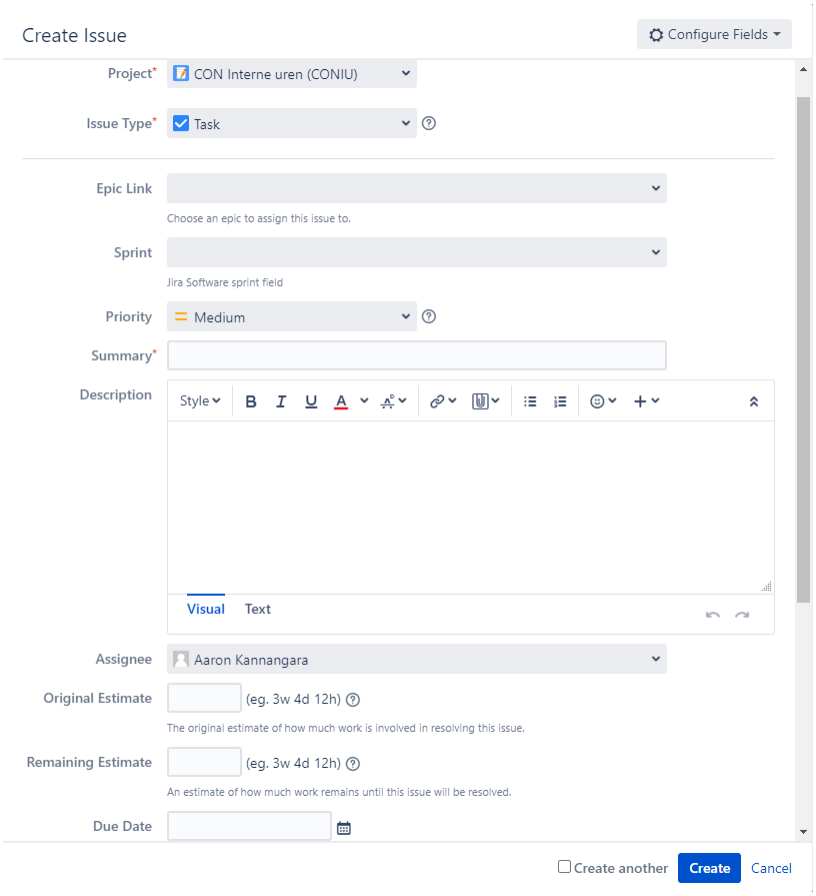



Figure 43: Example of Jira issue creation

# E Models generated by different discovery miners

In an attempt to reduce the file size of this document, all appendix images can be found in the projects Github repository[6]. For this reason only a description is provided with a direct link to that image in the repository.

Alpha miner model: Link.

Alpha+ miner model: Link.

Heuritics miner petrinet model with threshold 1.0: Link.

Heuritics miner petrinet model with threshold 0.99: Link.

Heuritics miner petrinet model with threshold 0.95: Link.

Heuritics miner petrinet model with threshold 0.90: Link.

Heuritics miner petrinet model with threshold 0.85: Link.

Heuritics miner petrinet model with threshold 0.75: Link.

Heuritics miner petrinet model with threshold 0.60: Link.

Heuritics miner petrinet model with threshold 0.5: Link.

Heuritics miner petrinet model with threshold 0.40: Link.

Heuritics miner petrinet model with threshold 0.30: Link.

Heuritics miner petrinet model with threshold 0.20: Link.

Heuritics miner petrinet model with threshold 0.10: Link.

Heuritics miner petrinet model with threshold values 0.979; 0.535 and 0.754. Generated using optimum threshold values found by Gaussian process: Link.

Inductive miner petrinet model with noise threshold 0.0: Link.

Inductive miner petrinet model with noise threshold 0.1: Link.

Inductive miner petrinet model with noise threshold 0.15: Link.

Inductive miner petrinet model with noise threshold 0.25: Link.

[6]Full details: https://github.com/akannangara/ProcessMiningThesis

Inductive process tree model with noise threshold 0.0: Link.

Inductive process tree model with noise threshold 0.1: Link.

Directly follows graph: Link.

# F Desired Workflow questionnaire, answers, and developed model

In an attempt to reduce the file size of this document, all appendix images can be found in the projects Github repository[7]. For this reason only a description is provided with a direct link to that image in the repository.

Desired workflow questionnaire to be filled in by team Scrum master, team lead and manager: Link.

Extract of answers to desired workflow questionnaire: Link.

Desired workflow model built using inductive miner with threshold 0.0: Link.

[7]Full details: https://github.com/akannangara/ProcessMiningThesis