Grip on software: understanding development progress of SCRUM sprints and backlogs

Helwerda, L.S.

Chapter 6

Discussion

Contributions of the Grip on Software research toward predictability of the SCRUM *software development process*

Abstract of Chapter 6

We finalize the Grip on Software research by reflecting on the problem statement, research questions, implementations and results discussed in earlier chapters. Particular consideration is given to the technical pipeline that made our studies possible. This includes components for agent-based data acquisition and database storage with a query template compiler for feature extraction. We also implemented state-of-the-art prediction models and validation methods that take temporal aspects of the data set into account. The research output is established by novel, integrated information visualizations.

We consider the design and architecture of our distributed research system to be part of our main contributions to the field of predictive software engineering. This is because they form a feasible approach to help explain how a SCRUM software development process is progressing. Our problem statement involves understanding how the analysis of events from this Agile process can be used to significantly improve predictability of sprint completion and backlog progression. We address these themes and problems through several research questions and sub-questions.

We conclude the research by answering these questions. The acquisition and storage of data relevant to the SCRUM process becomes reliable using our pipeline components, including MonetDB as a column-based relational database, for which we introduce a query compiler. For pattern recognition, we introduce several models for sprint workload classification and estimation as well as project backlog size estimation with reasonably accurate results. This analysis stimulates short-term and long-term planning efforts during the development process. Through information visualization, we provide understandable and usable visualizations which we evaluate through hybrid testing and survey assessments, showing promising adoption of several integrated visualizations that support analytical decision making and development ecosystem management.

The Grip on Software research project shows potential for future additions and follow-up studies, both in improving the quality of data and expected results for analyzing particular situations in SCRUM, but also through generalization toward other software development methods or reusability of implementations in future data-oriented research.

6.1 Retrospective

This chapter provides a recap of the Grip on Software research outlined in full detail across this thesis. The research focuses on understanding how to extract, analyze and explain events that take place during a software development process, which kinds of predictive algorithms are applicable to this kind of data and whether this allows teams to learn from factors that indicate the progress thus far. Specifically, we consider development cycles based on the SCRUM framework [6], with a total of 60 projects and teams from two Dutch governmental development organizations, ICTU and Wigo4it. Our focus of this chapter is to provide an overview of the results of this research.

As software development is a complex process, we have found that it is not easy to provide a single, encompassing conclusion about how development teams should conduct their work. In our research, we specifically focus on SCRUM, while remaining impartial on its efficacy compared to other software development methods. Keeping in mind the values posed by the Manifesto for Agile Software Development [3], we consider that there is no immediate need to outline the method's strengths and weaknesses, but rather to focus on potential usability. Rules that seem inherently contradictory on the surface in fact reinforce each other; for example, without processes and supporting systems, team interactions would become chaotic. In a similar spirit, additional predictability of the process—by planning ahead—should allow teams to respond to change.

Our main objective is to improve this predictability. This leads to a scientific approach in a situation where data is available, but scattered and not immediately ready for in-depth analysis. Through novel implementations of state-of-the-art models, we design a generalizable pipeline for secure data acquisition, analysis and explainable output. This is a cornerstone for providing accurate results through different methods of predicting short-term and long-term effort in SCRUM.

In this chapter, we address the results of the Grip on Software (GROS) research in different phases. In Section 6.1.1, we first discuss the GROS pipeline from a technical perspective. We then describe the main contributions per chapter in Section 6.1.2. Next, we reflect on the conclusions in Section 6.2, with the research problem addressed in Section 6.2.1. Specific points regarding the research questions are detailed in Section 6.2.2.

We wrap up this chapter and the body of this thesis with suggestions for potential future work that extends upon our study in Section 6.3. This includes possible research directions in Section 6.3.1 as well as generalizability of the approach—including applicability to other development methods and research endeavors—in Section 6.3.2.

6.1.1 Technical overview

Throughout our research, we have created and used a unified pipeline which processes data that we collect from software development projects. The steps that we perform in this pipeline are important to our needs and relevant to discuss on their own.

In Figure 6.1, we display a schematic overview of the components of the pipeline. In gray, we show some typical systems used in software development: an issue tracker to manage project backlogs with tasks like user stories, a version control system that holds code repositories and a quality metrics dashboard for checking coding practices. These are part of a software development ecosystem present at each organization. The red arrows denote our components that acquire data from these systems, import the data in a structured manner and extract features for further research purposes. Blue blocks in between the arrows depict intermediate, persistent artifacts of our pipeline. Finally, the green blocks represent the research output at the end of the pipeline.

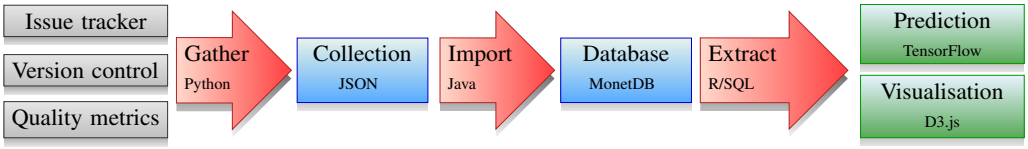

Figure 6.1: The full Grip on Software pipeline. The primary data sources (gray), processing components (red), intermediate artifacts (blue) and resultant components (green) are shown here. The components are constructed using various frameworks, modules and programming languages.

Each component is built with its own objectives in mind, with an overarching idea that the functionality of all the components is directed towards our data needs in the GROS research project. We describe the components in each chapter from a scientific usage perspective and clarify how we are able to employ the data in our studies after processing and transformation by the component. Even though the technical details of the components are not the primary subject of this thesis, they do pose a relevant topic when we augment the components with new techniques. Some of the novelty of the components is mentioned in Section 2.4.4 of Chapter 2. In Table 6.1, we indicate the chapters of this thesis and recap on what each of them describes. The component and intermediate artifacts are discussed from an applied, scientific point of view. In addition to these chapters, we briefly reflect on the technological improvements in this section.

PART	GATHER *Collection*	IMPORT *Database*	EXTRACT	PREDICTION	VISUALIZATION
Chapter 2	✓	≈	≈	≈	≈
Chapter 3	✗	✓	✓	✗	✗
Chapter 4	✗	✗	✓	✓	✗
Chapter 5	✗	✗	✓	✗	✓

Table 6.1: The chapters that each component or artifact of the pipeline is described in (✓) or not (✗). Some components are featured in multiple chapters or are only discussed briefly from a technical perspective (≈).

We summarize the technical work as part of our research, focusing again on the aspect of innovative approaches, e.g., in the field of data acquisition, pattern recognition and information visualization, in particular when it comes to handling data from Agile software development. We highlight specific aspects of each component in this section and describe why they are relevant for our research.

In the data acquisition component, our agent-based design supports the deployment of this portion of the pipeline in software development ecosystems that we encounter at different organizations, taking in account virtualization and networking. We configure the agents to access systems containing data from the development process. Each agent independently handles processing, including early encryption of personally identifiable information.

We model the collected data to fit in a consistent database model. We store the entities and relationships in a persistent, column-based tabular format in a MonetDB [38] instance. Through encrypted exports, we safely bring data together from multiple organizations at a central analysis location.

Feature extraction takes place using a set of query templates and reusable definitions that describe how to calculate velocity and other factors. We compile the templates into efficient SQL that MonetDB is able to further optimize. This also enables us to keep the templates agnostic to portions of our database model that are similar to each other. Configurations for each use case determines the relevant entities to select, based on their presence in the respective development ecosystems. This way, we obtain a consistent data set where sample features allow comparison and augmentation of models.

The prediction component provides novel methods that track the temporal data which was used for learning, while implementing state-of-the-art as well as well-founded methods, such as neural networks (NNs) and the analogy-based effort estimation algorithm (ABE), using TensorFlow [XI] for efficient, vectorized operations that are accelerated using GPU devices.

We finally produce multiple information visualizations accessible through an integrated dashboard plus several plugins for an issue tracker. We use a common basis of frameworks, such as D3.js [111], which allows us to build data-driven visualizations. We implement newly-designed controls for navigation, selection and full-page localization rendering. This leads us to the creation of various interactive visualizations. Each of them focuses on particular topics in software development. Meanwhile, their shared resources mean that it is easy to learn and switch between the individual views, leading to quicker adoption by stakeholders.

6.1.2 Main contributions

Our contributions to the scientific field extend beyond technical components of a pipeline for data acquisition in the Agile software development domain. One of our main objectives is to maintain a generalized approach in the Grip on Software research project. The modular design and data-driven method should work in other situations as well. We further consider the generalizability of the research in Section 6.3.2.

Through the individual sub-projects in GROS, we achieve and report on results regarding feasibility, performance, accuracy and adoption on progressive checkpoints of our research. This leads to the eventual result of providing traceable, interactive and explainable outcomes from predictive models. These models focus on classification and effort estimation of SCRUM sprint commitments as well as long-term product backlogs.

Before we consider the final results, we reflect on the advances of each chapter of this thesis on its own. We discussed the GROS data acquisition pipeline from a technical perspective in Section 6.1.1, but its introduction in Chapter 2 also concerns design patterns, modeling and performance, among others. The distributed, agent-based pipeline is a standalone contribution, but is inspired by work on distributed data systems, Petri nets and business analytics. We design the computational part based on these concepts. We also assert that a clear communication protocol is relevant, and introduce schema-based modeling to aid this. Meanwhile, we retain flexibility of placement of agents in the development ecosystem. Reusability of this component is also improved through continuous integration of development and documentation. The observed performance of the pipeline indicates its suitability for existing development platforms, without hindrance to the development team's usual process.

We continue our model-based principles in Chapter 3, where we introduce the GROS DB as our database system. Parts of the eventual database model that are thematically related due to their source system are explained. We further describe new relationships between entities from different systems. This also includes linking separate profiles when those profiles refer to the

same person, while preserving privacy and security of encrypted data. The introduction of a query template compilation system is followed by a performance analysis of the chosen system. The experiments show that our decision criteria—which led us to select MonetDB—turned out to provide a feasible, practical application.

Chapter 4 focuses on the pattern recognition problem, including extracting relevant features, training machine learning models and estimating short-term and long-term outcomes of SCRUM life cycles. This is the core of our research contribution. We discuss methods of selecting intuitive and essential feature sets using scoring, leading to a consolidated data set representing multiple organizations, teams, projects and sprint samples. Our new data set is split up into training, test and validation sets in a time-preserving manner. This approach allows models to learn from earlier samples while tuning and comparing accuracies based on later sprints. We introduce the tasks of sprint result classification/estimation and backlog size estimation, with models that are tuned for these purposes. The subsequent experiments with different data set sizes, parameters and scenarios show the effectiveness of applying these models in Agile software development planning.

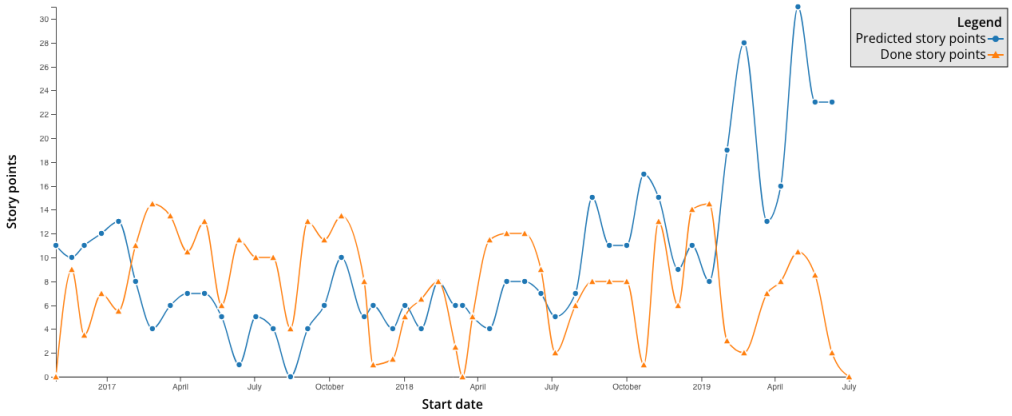

Figure 6.2: One of our information visualizations, the sprint report for a single project, showing effort estimation outcomes from predictions (blue line) compared to actual story points that the team completed (orange line). This is a way to visually represent analysis effort for experiments with limited training set sizes.

Our contributions culminate in the creation of information visualizations in Chapter 5. We make these visualizations available to the stakeholders as well as in anonymized form for further research. We produce visualizations for analytical decision support—the sprint report, prediction results, event timeline and leaderboard—as well as interactive views with an organizational ecosystem management theme, namely a collaboration graph, process flow chart, heat map and platform status dashboard. Three additional plugins are created for displaying product backlogs in new, interactive ways, namely predictive burndown charts, nested progression charts and relationship charts. Our work on extracting relevant patterns tightly connects to the visual format of the visualization hub, making the work more insightful. For example, we compare actual sprint outcomes with results from the analogy-based effort estimation across earlier sprints—by performing predictions with different training set sizes—through inclusion of both the feature and outcome in the sprint report visualization in Figure 6.2. Our methodical approach towards

improving the visualizations also leads to the conception of a hybrid assessment methods, in which we evaluate both automated tests regarding accessibility as well as input from surveys and interviews with stakeholders. This way, we validate whether our goals are met regarding the individual information visualizations.

6.2 Overall conclusion

We reflect upon our earlier introductions of the encompassing problem statement and specific research questions in this section. Based on our contributions—from a technical, scientific and organizational perspective—we conclude that our research implements state-of-the-art concepts, produces a novel data set with promising results and benefits multiple stakeholders that are active throughout the software development process.

We performed feature extraction, selection and scoring to obtain our representative, traceable and robust data set. We applied several models for the problems of classification and estimation of short-term and long-term planning in SCRUM processes. Throughout our experiments, we consider the performance of the models in this domain. We evaluate the models not only based on accuracy during the test and validation, but also by looking into effects of data set dimensions and various parameters, such as simulation scenarios.

Aside from the main problem statement in Section 6.2.1, we also discuss our specific research questions in Section 6.2.2 regarding separate focus areas of the Grip on Software research. This includes application of our pipeline in data acquisition and storage domains, the particularities of the pattern recognition analysis and finally the information visualization element.

6.2.1 Problem statement

The following problem statement is central to our research: “How can extraction and analysis of measurable events during a SCRUM software development process as well as other qualities of the product and team be used to significantly improve the predictability of practices employed at software development organizations?”

From the beginning of our research, we looked into what data would be useful to explain factors that contribute to the success of a SCRUM software development cycle. We focus on data that is made available through the use of project trackers, development platforms, quality checkers and other systems in software development. There is a constant generation of data in this development ecosystem. Due to this, team members and other stakeholders are not equipped to inspect all of the data, manually or otherwise. Thus, it is difficult to maintain an overview of the process or to spot specific problems. This is why we determine that automated data acquisition and analysis through machine learning benefits the development process as a whole.

Due to the repetitive nature of SCRUM, our underlying hypothesis is that the data exhibits relevant patterns with emerging properties of both short-term SCRUM sprints and long-term product backlogs. This would allow us to derive consistent and compact definitions for machine learning features, thus abstracting away from potential behavioral differences between teams and organizations. Meanwhile, we retain the characteristics that describe the progress in an unbiased manner, including potential alternative properties for effort estimation like number of comments

and code commits. This yields a data set that is suitable for solving classification and estimation problems, contributing to the predictability aspect of our research question.

By focusing on collecting and analyzing the measurable events of SCRUM, we ensure that the research remains objective and explainable. We note that some input is based on expert judgment, for example the assignment of story points to the work items planned for a sprint. Still, we are able to perform comparative analysis of those attributes through various feature extraction methods. Using novel approaches to existing software engineering metrics regarding sprint velocity, we obtain to a form of "exchange rate" that streamlines the input data across teams. We take much care in tracking the origin of features used in our data set to fulfill our explainability objective. When we provide our research output to stakeholders, they are able to find out what the features mean. For some estimation models, we also indicate which sprints were similar to the one being estimated. On the other hand, personal data is encrypted early and only plays a role in properly modeling relations between user profiles, while the eventual data set itself is anonymized.

An important factor in increasing the impact of the Grip on Software research project is to move beyond just providing data. Even if metrics are annotated or provided in graphical form, more steps are necessary to ensure that end users learn from observations and results through interaction. We provide different visual forms of various parts of the curated data, which allows stakeholders to view representations of focus areas that are part the development ecosystem. By interacting with these visualizations, they are able to make decisions based on configurable reports that focus on relevant factors. Transformations based on this interactivity reveal how situations compare to each other and accentuate differences between teams and organizations. At the same time, we decrease the cognitive load when the user switches between visualizations through common control elements—interacted with through simple gestures—in the interface.

6.2.2 Research questions

From our principal problem statement, we deduced several specific research questions. Each of these three questions focuses on a different facet of the GROS research, namely (1) data acquisition as well as storage, (2) pattern recognition and (3) information visualization. These themes also describe the natural flow from data extraction to usable research output while preserving explainability. The same process is embedded in our component-based pipeline shown in Section 6.1.1. Similarly, our thesis is constructed around these three research questions, each with four sub-questions, with the first research question split across two chapters. In this section, we consider each question and how we answered them in those questions, by summarizing the results obtained for each segment.

Data acquisition and storage (Chapters 2 and 3)

RQ1 How can we reliably collect data regarding SCRUM software development practices and consolidate the resulting artifacts inside a central database that constantly grows and allows adaptable queries?

The first question that we considered focuses on the technical aspect of the research. We address the problem of collecting and safeguarding large-scale, incremental data streams by means of our contribution of the pipeline. In particular, the data acquisition and database components perform

these tasks and feed the necessary input for solving the remaining objectives regarding prediction and visualization. Both steps mentioned in the question, i.e., data acquisition and consolidation, are important in this research field for a proper, curated approach. We design and construct our pipeline according to a structured, modular method for performing common quantitative and qualitative tasks to make the research work in diverse networks of systems.

We split up this question into several points regarding the two components, in order to further understand how to approach data acquisition and storage.

RQ1a Based on relevant design principles from distributed data systems and agent-based networking, how do we design a data acquisition pipeline applied to multiple organizational ecosystems?

We compose an agent-based network design, in which our distributed systems regularly check for updates of data in issue trackers, code repositories, quality dashboard and other systems that they are configured to access. This implementation works well in organizations that have multiple projects, each with their own virtual or physical networks. An agent-specific configurator allows setting the proper credentials and access settings for each data source. We implement specific technologies to access various APIs to obtain updated source-specific data, convert the data to standardized forms based on JSON exchange specification schemas and upload the new data via SSH to a centralized controller for further processing. We perform experiments of this agent-based setup in multiple projects using different development ecosystems to ensure that the activity of the pipeline components did not impact the performance of their usual process.

RQ1b Which objectives related to inspection, curation and disclosure do we achieve with a data acquisition pipeline when taking our formulated requirements in mind?

During the data collection, we monitor the incoming data through inspection of intermediate artifacts and query-based reporting of the database status. Additionally, we deploy a status dashboard that displays health information of all the distributed agents and collects logging information at this centralized location. By sampling situations from the database and looking for uncommon situations, we perform curation of the data. One example is an adjustment of how story points (SP) are counted when nonstandard values are filled in by developers to indicate that the story is not yet ready for development. Personal data is encrypted as soon as possible, while still allowing curation of duplicate accounts through links as explained for **RQ1c** hereafter.

Finally, disclosure of the data to stakeholders is provided by the later pipeline components for pattern recognition and information visualization, which integrate with the database component, making use of template-based feature extraction and data aggregation, respectively. This allows the data acquisition component to restrict its scope to just the input side of the pipeline, with the database acting as an intermediary for further access of the results. This modularity is an important factor for potential reuse of the pipeline and helps with further dissemination of all artifacts produced during the data processing.

RQ1c How can we model the relationships between different data artifacts acquired from dynamic systems regarding SCRUM software development in order to properly deduce information about their state during different sprints?

The database component is realized by MonetDB, a column-based relational database management system (RDBMS), which is well-suited for large-scale analytical information processing. Entities are stored in the database as rows that are part of tables, with properties in their columns. Relations between each entity are either additional properties or separate tables, depending on the type and cardinality of the relation. Using a column-based storage, accessing details of many entities at once is highly performant.

Based on the entities and relationships acquired from the source system, we model additional references. These link entities that already existed in the same source system as well as between different parts of the schema. Our focus here is increasing the descriptiveness of our main entities, oriented around projects and sprints. Sometimes, this is accomplished by performing inference of events that took place between a sprint's start and end dates. This allows us to uncover what has happened in a sprint through queries based on well-defined time sequences.

We further address privacy concerns during linking of the same person that exists across systems in the development ecosystem. We use project-specific and global encryption keys for personal data, allowing us to relate the same entity with different profiles without being able to read its properties—unless the original data was already available. The encryption keys remain at the organization where the sensitive data is acquired, with only the pseudonyms available during our analysis.

RQ1d Which technical challenges are relevant when deploying a database component as part of a pipeline for multiple organizational ecosystems?

The architecture of the database component is suited for different development ecosystems. We set it up as part of the pipeline at the development platform of a participating organization, while we also maintain a central instance for research purposes. The resource usage of the component is limited through the implementation of batched update statements during imports as well as optimizations of queries.

Meanwhile, we increase the flexibility of the feature extraction queries through the introduction of a template compiler. The compiler dynamically adapts the queries to the usage of systems by the organization. This means that we obtain similar sample data for machine learning features regardless of the type of project management system that tracks stories, work items and other tasks. Currently, both Jira and Azure DevOps—formerly known as Team Foundation Server (TFS) and Visual Studio Team Services (VSTS)—are fully supported in this way, while entities from GitHub and GitLab have been modeled likewise. Despite limited use of the issue tracking functionality of these two code review systems at the two organizations, their modeling in our database demonstrates potential for generalization of the technical approach.

Experimental results show that both our manual improvements to the template queries and optimizations provided by a hot-started MonetDB instance help with speeding up the queries. This enables us to perform rapid, frequent updates of data for the pattern recognition and information visualization purposes later on in the pipeline.

Pattern recognition (Chapter 4)

RQ2 How can we improve the predictability of SCRUM software development practices, specifically the progress of sprints and backlogs—based on analysis of data selected from the development process—and how do we validate our approach?

Our second research question concerns the core of our scientific approach, namely the machine learning models and related pattern recognition methods. From our results, it becomes clear whether each method is feasible, explainable and usable by stakeholders. We present adjustments and improvements of existing models to work on the specific problems of predicting outcomes of sprints and progression of work on backlogs. We extract features for our data set and select groups of them that hold descriptive qualities of the sprints that they originate from. By applying common means of validation on our models, we observe the accuracy of the results and understand their behavior. Stakeholders are then able to make use of these results, supported by explanations on how the machine learning models arrive at the conclusion of the provided labels. These then help with answering common questions in SCRUM, such as how many story points should be planned for the upcoming sprint or when the backlog will be completed—on time, preferably.

RQ2a Which features can we select based on ongoing data from the software development process that are most indicative of the progress?

Using our query compiler, we extract features regarding progression of SCRUM sprints of multiple projects from two separate organizations in a standardized effort. The features are scored individually and as subsets through external relevance metrics as well as analogies provided by one of the models. Tuning of hyperparameters is then possible to improve accuracy and confidence of the results. The selected features are related to team size and experience, sprint and backlog planning, code version control and quality metrics. Section 4.4.1 of Chapter 4 lists the selected features.

RQ2b What kinds of learning algorithms can we introduce to this problem, which learn from these features and provide feedback on what kinds of decisions can be taken?

We apply several machine learning models and statistical approaches to our problems of sprint outcomes and backlog sizes. These include state-of-the-art and well-known models, such as deep neural networks (DNN). Some methods have been proven successful in other fields, while others originate from software development already, e.g., analogy-based effort estimation (ABE). The neural networks are applied in this problem to provide a classification as to whether a sprint is at risk of finishing with non-completed stories. ABE goes a step further, synthesizing a number of story points based on similar sprint samples. We also perform estimation of future backlog sizes with two simulations, namely a linear regression algorithm and a Monte Carlo method, using three different scenarios for mutations on the backlog. Together, these models predict the progress in short-term and long-term durations, respectively in sprint commitments and backlog milestone planning. This encompasses the guidelines and provisions of the SCRUM framework, extending it with improved predictability without hindrance to self-managed task commitments.

RQ2c How do we predict the likelihood of timely and properly finishing a currently-running sprint or a longer-running period aimed at resolving a product backlog within a development project, even before it has started?

Using the DNN and ABE models mentioned in our answer to **RQ2b**, we predict the outcomes of sprints, both before a sprint has started as well as during it. The neural network provides not only a classification but also an indication of how reliable the prediction is according to the model itself. Analogies further help with understanding the second model's effort estimation, as they form the basis of the number of story points that are likely to be completed. This allows these models to help development teams through their sprint tasks and meetings—from refinement until retrospective—by providing relevant insights in their process.

For longer periods of time, the backlog size estimations demonstrate potential end dates of completing a certain collective workload by simulating changes to the product backlog across multiple future sprints. Different scenarios provide better detail on the potential changes, with potential for learning from other projects. This has allowed some of our stakeholders, particularly software delivery managers, Product Owners and analysts part of development teams, to determine whether a specific milestone is reachable. Even teams that are just starting with planning stories at the beginning of a development project or are interested in work planning for subsets of their backlog benefit from these algorithms.

RQ2d How do we validate that the predictions and recommendations are within our expectations and based on relevant, explainable factors?

Our research shows potential for the practical applications of the models, but we do not omit well-founded results regarding their accuracy and explainability. We validate the models, using the F1 score and similar metrics appropriate to each model, while comparing the sprint classification and estimation to a baseline guess of taking the previous sprint's outcome as the prediction.

Both the neural network—with an F1 score of 89.98%—and the analogy-based effort estimation, where 88.6% of the estimations of one organization are within a margin of 25%, performed better than the baseline on our temporal data set. We note that some variance is observed when the data set size is adjusted. As for explainability, we consider that feature scoring and built-in tuning of some of the models has helped to select representative subsets of features that are suitable to solving the problem at hand and helpful for stakeholders to track.

For backlog size estimations, the most likely progression is provided by the Monte Carlo method that simulates all potential changes on the backlog. A linear regression model too much variance. A scenario where only the velocity is accounted for could not properly simulate sudden changes to the backlog's scope. We validated these results by taking a fraction of previous progression of the product backlog, measured at the end of each sprint, as our data set and comparing the model's estimations to the actual progression. We measured the mean error as well as the largest deviations. We further analyzed the behavior of our backlog estimation models per project, taking into account differences in backlog sizes in each of them. Finally, we validated the distributions generated by the Monte Carlo simulations, to determine if the scenarios remain normalized despite the introduction of additional factors.

Information visualization (Chapter 5)

RQ3 How can we effectively introduce visual representations of results and recommendations from our analysis of data collected from the SCRUM development process to the involved parties?

The third and final research question of the Grip on Software project concerns the presentation of data to relevant stakeholders. This shifts our focus from purely analysis, where solely achieving an acceptable accuracy is already a plentiful outcome. For this research question, we consider how the end user benefits from the collection of previously unlinked entities in novel ways. This includes the results from our pattern recognition methods, but through an integrated visualization hub, we intuitively zoom in to various aspects of the breadth of information to showcase the possibilities that the GROS pipeline provides.

RQ3a Which concepts and goals are relevant when designing information visualizations for patterns analyzed from a software development process?

As part of our introduction of a comprehensive visual summary of the results regarding the outcomes of SCRUM sprints, among others, we outline the necessary objectives, concepts and goals regarding visualization for Agile software development, information visualization in general and the structure of our specific dashboard implementation, respectively. This includes stating the purposes regarding display of temporal, dynamic data, making data easily comparable, gaining insights and sparking new ideas.

We consider existing flows of transforming data into visual forms that are rendered in an interactive view. We map this method upon the components of our pipeline as well as the cognitive process of gaining knowledge from descriptive data. We find that an integrated approach—both in our technical component as well as regarding the process of bringing the visualizations to the involved persons—and a clearly-outlined purpose help make the data become information that is useful for stakeholders to learn from. This helps boost delivery of the product and promotes new ways of thinking in the development process.

RQ3b How do we integrate results from predictive models within existing development practices?

Our visualization hub consists of a dashboard which provides access to various information visualizations, with a similar technical and interactive implementation. The visualizations are grouped by focus area, with two main categories: analytical decision support and software development ecosystem management. The former group includes visualizations for reporting on progress of teams across earlier sprints in various formats. One of the visualizations is arranged as a distinct location for the results of our sprint predictions. A clear layout makes the main results prominent, while allowing further scrutiny through detailed tables of features, model configurations and—depending on the model—additional metrics or a list of similar sprints.

Further disclosure is achieved through an API which allows external systems to integrate with the results. One implementation to display the prediction result in a quality report was made for this purpose, helping developers see whether they should consider planning a different number of story points for the next sprint. The results from our analysis of sprint predictions across different training set sizes as well as the backlog size estimations are available in the sprint report visualization, which lets users freely make projections for planning purposes. Additionally, the Jira plugins build upon the analysis and results from the backlog size prediction, by allowing users to view a breakdown of a product backlog across time, including the future sprints simulated by the Monte Carlo method.

RQ3c Which effects do the introduction of results from predictive models and other intermediate analyses have on the development process, validated across multiple ongoing projects?

We present the information visualizations to potential users during surveys and interviews. These stakeholders include developers, quality engineers and product owners. During our study, we assess the usability of the visualization through various metrics, while asking the users how they would consider using the information contained in the visualization during their daily work. Whereas initial surveys for a limited number of visualizations showed mediocre potential usage, we later performed further interviews which indicated a significant adoption, especially regarding the use of the sprint report. Various teams use this visualization to learn from earlier sprints by selecting primary indicators that fit their team the most. This feedback was helpful for us, by confirming whether selected feature sets that we applied during the predictions are indeed relevant.

The output of the prediction models has been further integrated with the visualizations through as well as existing quality reports. In particular, earlier evaluation showed that there was a need to easily view attributes of previous, current and future sprints, thus these attributes plus the backlog size estimations were made available in the sprint report. This visualization is used by product owners and software delivery managers to gain an overview and to plan full backlogs, with planning based on milestones supported tentatively. Our predictions are considered a beneficial addition to the existing workflow. The use of predictions as a meta-metric gains adoption with an explainable backtrace to the contributing factors. By creating plugins for Jira, we brought such information back to more teams.

RQ3d What is the overall assessment of the proposed information visualizations, considering automated measurements for usability and the adoption according to interviews and surveys?

We combine integration test results with user adoption surveys to produce an assessment of the information visualizations. In general, the visualization hub is found to be usable. This is boosted by considerations on, e.g., colors, glyphs and consistent controls, but also wider topics like traceability of data, mobile-friendly design and accessibility. This allows adoption by different roles in the software development organizations, regardless of technological background or impairments such as color blindness.

By focusing our visualizations on topics at an organizational and project-specific level, both in the analytical and ecosystem management domains, they are well-received and align with

regulations regarding privacy. Finally, we remark that there is still more to explore for information visualization in software development, particularly when it comes to planning and product development, to further enhance this process.

6.3 Future work

A primary objective of our research is exploring the feasibility and performance of several pattern recognition methods in predicting potential effort in SCRUM sprints and entire project backlogs. This research has shown that we are able to acquire a representative data set to obtain accurate, explainable predictions and deliver relevant information back to the stakeholders by creating interactive, integrated visualizations. Therefore, we deduce from our answers of the research questions that we meet many of our objectives in several sub-topics, including technical implementations, data modeling and extraction, machine learning and eventual presentation.

However, there is always more research to be done, especially in the field of predictive software development. While we have provided answers to our research questions, there still remain unknowns in this domain regarding other software development methods or even other processes part of the SCRUM framework.

In other chapters, we briefly discussed possible improvements and methods to be considered. We are hopeful that our work functions as a viable means of continuing research. For Grip on Software in particular, we hypothesize that further integration of data-driven predictions in software development helps improve the process as a whole, by letting involved people find more emerging patterns. This process allow learning from past situations, current recommendations and future projections in the same place. In the end, this research makes repetitive events—which would otherwise feel ceremonial to developers—more enriched through the addition of easy-to-use visual results.

Through generalization, some of our contributions become beneficial when applied in other focus areas. Our approach toward designing and implementing modular pipeline components is translatable to other exploratory data-driven analysis. This technical foundation is available for others to build upon, since similar challenges and objectives regarding collecting, structuring, selecting, analyzing and representing large amounts of data exist elsewhere too.

Our vision of prospective research based on this thesis is therefore twofold. We consider the particular methods and models that build upon our data set in Section 6.3.1, with some limited speculation on what we imagine the end result to look like. To address possibilities beyond our current scope, we consider applicability to other software development methods and research studies in general in Section 6.3.2.

6.3.1 Further research

In several chapters, we discuss our intermediate conclusions for the topics at hand there, with suggestions regarding possible avenues of further research as well. We expand upon the earlier contemplations with several promising applications of our data in improving the predictability of SCRUM software development in more ways. We considered some of these sub-problems to become a part of our core research—with some initial work available for them as well—but some topics simply strayed from the “narrative arc” that we sought to maintain in our work.

In Chapter 4, we perform classification and estimation of SCRUM sprint outcomes and full project backlogs. One possibility is to extend this approach toward other time ranges and subsets of the backlogs. A proof of concept, where we determine when work for a specific milestone would be completed, is also shown in Section 4.7.2. Ideally, we would be able to consider how much time an early backlog will roughly take to complete, even when the development team has not awarded story points to most of the work items.

We also mentioned other models for improved accuracy and explainability. One option would be the application of Long Short-Term Memory (LSTM), a model which includes contextual information about nearby samples during training. This has shown applications in linguistic processing and other consecutive data. Because our data is temporal in nature, models that understand the relations with earlier events—e.g., when stories are moved from one sprint to the current or future sprint—are well-suited in providing more thorough labels, clusters or estimations for the samples in the GROS data set.

Based on the model of our database, we are able to extract features for other entities than just sprints or projects. Through our curation with the query template compiler, we extend our understanding to the level of teams, which benefits organizations where teams work on multiple projects or share responsibility on a larger project with multiple teams by developing individual components. Some of such multi-project teams still need some additional curation to reliably combine data from simultaneous sprints, including how story points scale the entire backlog. Project-level and team-level predictions then become more comparable.

More in-depth predictive analysis is obtainable by defining features for each story, code change or quality check, leading to an even larger data set that provides novel classification opportunities. One possible application is to provide an *early warning* to developers when something seems wrong with a specific metric, such as a missing field in a work item or an unexpected decrease in quality.

Due to the continuous nature of code quality measurements and development platform stability, we also consider *anomaly detection* to be a beneficial technique, where rare situations are found and presented for further analysis on why they take place. This leads to potential new features for use in other machine learning problems, allow early warning on its own or be used as a hint for tracing back why a feature is different than normal in the first place.

Other ways to augment our data set is to include external data, such as weather or traffic information, in order to find possible reasons for differences in development throughput day by day. Although we focused on objective data available in the software development ecosystem, we should also keep in mind the human aspect, even in our data set. For example, a developer's mood likely affects the quality and quantity of code written by that developer. It is conceivable that *sentiment analysis* on comments left in issue trackers and version control systems provides an indication of how well a sprint is going. However, more intrusive approaches—such as recording and transcribing all of the team's meetings—might encroach too much of the sensitive nature of such group events, even if the output is anonymized before combining with our current, privacy-aware pipeline.

Instead, we should focus on implementing more means of explainability in our predictive models, along with other improvements in terms of feature extraction and types of model outputs, including inherent accuracy metrics. In our case, this approach advances integration of results in the SCRUM workflow, with less potential for misunderstanding. Better tracing of where a neural network's activation comes from, regardless of the network size, leads to improvements of the perceived reliability of the algorithm, which is important to our stakeholders.

Other implementation decisions would be on-line re-runs of backlog estimations, such that users are able to try different parameters and subsets for planning. This functionality is then available through a visualization shown in Chapter 5, in particular the sprint report or through a plugin for Jira. We wish to further explore how information visualization is applicable in the Agile software development domain, with more integration of the information visualizations. This leads to a single entry point where it is possible to dive deep into specific details, while retaining a simple front-end for obtaining overviews regarding progress. Each *zoom* level then has its own presentation format applicable to the information visible there, such as tables, charts and network graph layouts.

6.3.2 Generalizability

The focus of the Grip on Software research project has been on software development processes that use the SCRUM framework. However, it is possible to expand the scope to include other areas of interest. It would be interesting to compare other software development methods to ascertain whether similar patterns are found there. Due to the repetitive nature of SCRUM, we are able to create a consistent data set of sprints as our feature-rich samples. Mapping such specific events to other frameworks is a challenge with a potential high reward, as a more broad model leads to wider applicability of the results of this extended research.

Certainly, the most sensible development methods to consider would be other Agile processes, such as extreme programming, Kanban or Lean, which share similarities with SCRUM. It is feasible that our approach also works when corresponding entities are not easily mapped. We would then focus on other levels of detail in work volume, such as milestones or individual issue items. At the organizations in our study, we also collected information on support teams. Some teams do not use SCRUM—or only for a portion of the workload—but we still provided some reports to them regarding work done in specific time frames.

Given that the two organizations are both Dutch IT implementation agencies for governments, one major opportunity for advancement is to include more diverse development companies in our data set. Each organization has its own working environment, including differences in the application of SCRUM among their development teams. Especially multinational businesses with colocated teams have widely contrasting concerns and challenges compared to a tightly-connected, local organization. The effect of developing products for-profit—instead of based on government budget—may also play a role in team workload. Overall, more breadth of data from various organizational instances leads to better models, but also more research hypotheses that we are able to test.

The technical infrastructure should be able to handle more diverse data sources. We have shown that our database model is able to handle entities that originate from different systems and provide them in a consistent format through our query template compiler. Development ecosystems with other platforms, trackers and collaboration software are easily added, as is the case for GitHub and GitLab, for example.

In general, our combined approach toward data acquisition, storage, modeling, analysis and visualization are available as building blocks for more studies in this field and even for other data-oriented research. The modular nature of the components allows adjusting the purpose of the pipeline by replacing or extending parts of it. While there is a focus on software development data, in theory the use of schema-based data exchange allows extensive abstraction of the type of incoming data.

The openness of data formats also allows better dissemination of the results of our research. Through a consistent API specification, we provide both the data set used specifically for the model as well as detailed information on model configuration and validation outcomes. This encourages not only reuse of the input data, model and labels for further study, but also sets an example on how transparency regarding analysis parameters helps to create feedback loops. Integration of such details in an intelligible manner in information visualization helps to foster interaction and discussion that exceed initial expectations. A human-centered approach in visualization leads to new possibilities, in the software ecosystem and beyond.