Master Thesis Proposal: Risk detection and risk control through sprint backlog analysis

Sytse Groenwold
13083228
sytse.groenwold@student.uva.nl

Zhiming Zhao
Supervisor
z.zhao@uva.nl

TBD
Examiner
email

ABSTRACT

Abstract comes here in the final stage of writing.

KEYWORDS

Keywords come here in the final stage of writing.

ACM Reference Format:
Sytse Groenwold, Zhiming Zhao, and TBD. 2022. Master Thesis Proposal: Risk detection and risk control through sprint backlog analysis. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In software development, there are several risk events that can decrease product quality, delay delivery or even cause the project to fail[2]. Tangible examples of these events are budget overruns and losing personnel, while more subtle ones include a lack of clearly defining work to be done or team members being unaligned. (Early) detection (i.e. risk assessment) of such risks and controlling them effectively (i.e. risk control) is crucial for the success of a project[3].

The Agile development methodologies have demonstrated their advantages over traditional methods in reducing time to market, overall product quality and a closer alignment to business needs[5]. In Scrum, this is achieved through an iterative process over a short period of time; these iterations are called sprints. The product manager creates a backlog of user stories, which are the business requirements translated into pieces work for the development team. During a sprint, the team works on a selected number of these user stories and afterwards they reflect on both the work done and the collaboration within the team. Despite these improved practices, Agile methodologies can still suffer from risks: unclear user stories, inefficient decomposition, and scheduling of the sprint backlogs. Determining those risk through analysis of the user stories in the product backlog is mostly done by human experts.

In a previous study, we sought to automate this process, by creating an machine learning model based on previous works and ensembling those models.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, July 2017, Washington, DC, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

While those results show promise for a machine learning model to detect risks, improvement is needed to make the predictions more robust, especially to achieve more consistent scores between different datasets. We determined that incorporating the sprint plannings, instead of only analysing the individual user stories, could be a valuable addition to the model parameters in achieving this goal. In this study we plan to do that, and additionally investigate a way to single-out the specific risks that are inside a sprint backlog, so that teams can exercise risk control before they affect the team output.

Among the different software packages used by Scrum adaptors to track user stories, Jira is by far the most used one, having a large margin over other comparable services, according to slintel[1], Datalyze[2], and Enlyft[3]. It is however still under represented in software engineering research, according to Montgomery et al.[7]; for this reason they published their dataset with the most extensive collection of publicly available Jira repositories to date.

1.1 Research question

Q1. To what extend can the inclusion of sprint backlog in issue analysis improve risk assessment within software teams using Scrum and **Q2**. which features of such a model are most impactful for both risk assessment and risk control?

1.2 Sub questions

The sub questions are:

Q1.1: How many publicly available Jira repositories have their sprint data available?

Q1.2: Is it possible to include this data as a single feature into existing models, or would a separate model added to the ensembling method perform better?

Q2.1: Is the sprint data a more direct indicator for risk assessment than the issue fields?

Q2.2: Will adding/removing certain issue from a sprint work lead to a lesser amount of overridden sprints?

2 EXPLORATORY DATA ANALYSIS

Data sets extracted from the Jira issue tracking software (IST) will be used in this study, because of its high adaption rate, wide-scale public availability, and high number of data fields on each issue. The data sets that have been considered for this study are publicly available data sets which mostly come from GitHub accounts of other researchers whom have gathered them for previous studies. The dataset used in this study is the one created by Montgomery

[1] https://www.slintel.com/tech/bug-and-issue-tracking/atlassian-jira-market-share
[2] https://www.datanyze.com/market-share/project-management–217/jira-market-share
[3] https://enlyft.com/tech/products/atlassian-jira

et al.[7], because it is has the most projects, is the most recent and has enhanced data. In case this dataset appears to be insufficient, there will be an option to gather a data set from an organisation. This dataset will also be considered for a validation set.

Figure 1: Jira MongoDB database scheme

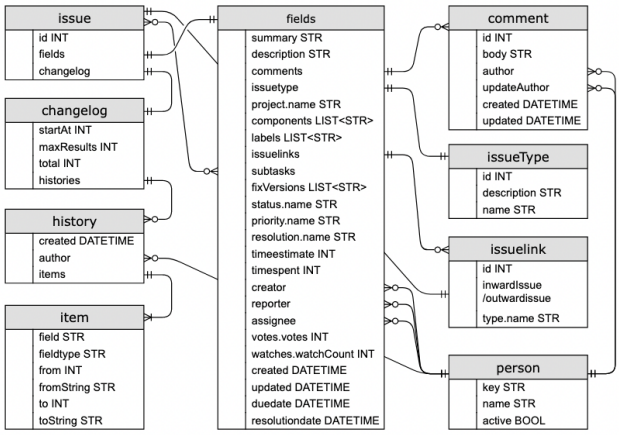


Figure 2: ERD

Figure 1 describes the entity relationship diagram (ERD) for the data set as it modelled inside the document data store MongoDB. Due to nested values in document-oriented data stores, the ERD is a simplified depiction of the real data structure, which can include various values for the historical changes of the issues, as well as the different levels of links between issues. The primary documents are issues. The two main documents nested inside each issue, are "fields" and "changelog". The fields are are the current values of the attributes inside an issue, as described in Table 2. The changelog are historical values of these fields, as well as some meta data, i.e. number of times changed and a date and time of the change.

The dataset is very extensive and complete and holds several types of data: text, datetimes, nominal and ordinal numerical data, linked network data and historical versions of all nearly every field. The fields are a mix of singular values, lists and dictionaries.

The most important fields of the data set arguably is the enhanced mapped data of links and types. Not only does this normalise parts of the data and reduce processing requirements, but it also adds the possibility of using these custom fields. Choetkiertikul et al. [2] have showed the value of this type of data, therefore this study continues in that same direction.

The data set consist of 16 public Jira repositories with a total of 1822 different project, containing 2.7 million issues. These issues contain historical data of 32 million changes, 9 million comments and 1 million issue links of various natures. Table 1 shows the complete overview of this data: the year the projects started, the number of issues, number of documented and used issue types, number of documented and used link types, number of changes per issue, number of comments per issue, and number of unique projects.

Table 1: Description of fields in Jira issues work items.

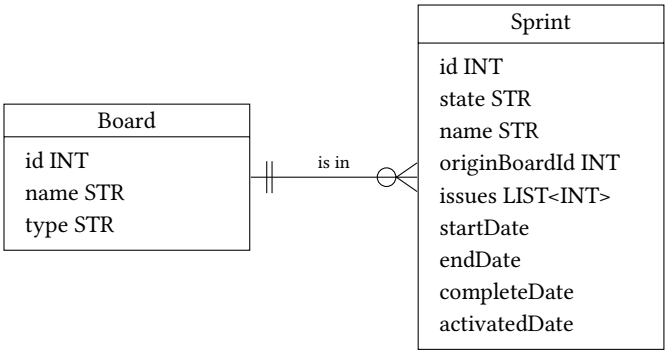
Issue field	Description
	Issues
summary	A brief summary of the issue.
description	A longer, detailed description of the issue.
comments	Community discussion on each issue.
issuetype	The issue purpose within the organisation.
project	The parent project issue belongs to.
components	Project component(s) to which issue relates.
labels	Label(s) to which this issue relates.
issuelinks	A list of links to related issues.
subtasks	Sub-issues to this issue (one level deep).
fixVersions	Project version in which the issue is fixed.
status	The stage the issue is at.
priority	The issue importance in relation to others.
resolution	A record of the issue's resolution/closure.
timeestimate	Estimated amount of time required to resolve.
timespent	Amount of time spent working on this issue.
creator	The person who created the issue.
reporter	The person who found/reported the issue.
assignee	The person responsible to resolve the issue.
votes	Number of people who want this issue addressed.
watches	Number of people receiving updates
created	Time and date this issue was entered into Jira.
updated	Time and date this issue was last edited.
resolutiondate	Time and date this issue was resolved.
duedate	Time and date this issue is scheduled to be completed.

Table 2: Dataset of Jira repositories

	Sum	Max	Min	Median	Std Dev	
Born	-	2016	2000	TBD	TBD	
Issues	2,686,282	1,014,926	1867	59,853	251,973	
DIT	485	122	1	18	31	
UIT	352	55	2	14	16	
Links	978,223	264,108	46	15,898	81,744	*
DLT	180	23	6*	10	6	
ULT	169	20	4	10	5	
Ch/I	-	30	3	12	6	
Co/I	-	8	1*	3	2	
UP	1822	657	3	37	178	

The two no longer available repositories do not have these values
Column names: Documented Issue Types (DIT); Used Issue Types (UIT); Documented Link Types (DLT); Used Link Types (ULT); Changes per Issue (Ch/I); Comments per Issue (Co/I); Unique Projects (UP)

Figure 3: ERD of the boards and sprints Jira documents



Jira being a customizable IST explains why there are more DITs and DLTs than UIT and ULT. It is possible that some are created and are not yet or no longer used. Custom fields are not directly usable in a possible model, due to them not being normalised. The authors of the data set therefore enhanced the data set by mapping the links and the types to a curated number of options. This enhancement allows for use of these custom fields, better comparison between different Jira repositories, and projects within those repositories.

The minimum number of changes on any of the nearly 2.7 million issues is 3 and the median is 12. If these are considered by the model, it would mean that the data set holds close to a magnitude more data than any of the previous researches, where nobody used this historical data.

The sprint iterations documents and boards documents retrieved from Jira hold less fields and can be seen in figure 3. The boards are merely described in this diagram, due to each sprint needing to be contained within a board and the sprints themselves are sourced through iterating over each existing board. The sprints do hold not hold many fields eligible to be used for analysis: the different dates allow to calculate the duration of the sprint and a difference between endDate and completeDate indicate an overrun or the administrative action of closing the sprint being done later.

3 RELATED WORK

3.1 Agile software development

Software development is known to face issues with delivering expected quality and overrun schedules and costs.[5] Agile methodologies attempt to combat these issues through shorter delivery windows by iterative development, focusing on interaction and communication between all people involved (team members, business, stakeholders and end-users), and reducing the size of the intermediate results. This leads to teams being more capable to react to changing requirements and prevent waste of time and resources.[3]

The most widespread used Agile methodology is Scrum[4]. Following this methodology, teams work on *user stories*, which are the system requirements. These user stories are created by the *product owner* and based on the requirements as defined by the business, stakeholders, and the end-users. Together with the team, the product owner can *refine* the stories and assign it *story points*, which are arbitrary numbers that signal the amount of effort (not time) it takes to complete. The user stories are prioritized and stored in the *product backlog* until selected. The team works in *iterations* of work called *sprints*, which are *timeboxed* to usually 2-4 weeks long. At the beginning of a sprint, the teams selects user stories from the product backlog to work on, which they intend to finish within that sprint. User stories can therefor never take longer than the duration of sprint; if they do, they need to be split up into smaller ones. User stories inside the sprint backlog do not change during the cycle. Every day there is a *standup*, where each team member shares their progress on the user stories and adjust if necessary to complete the work. At the end of the sprint, which is *timeboxed*, the team holds the *review* where the outcomes are discussed and during the *retrospective* they gather feedback to use during the next cycle. If some work is not done, it automatically rolls over to the next sprint, although this must be avoided.

Despite these benefits, a study in 2014 showed that many adaptors of the Scrum methodology (in Norway) still address theirs risks in their old, traditional approaches.[8] Additionally, even if some problems are solved through adaptation of Scrum, new risks can possibly be introduced, such as risks in the sprint planning, risks in the code base and technologies used, an increased separation between the development and the operational work, and increased impact of technical debt[6][9].

3.2 Risks assessment and control

Meerman[1] looked at individual issues and used some of the default features of a Jira isssue: summary, description, status, type, labels, project, assignee, created, updated, resolution date, points, urgency, severity and sprint. They made use of ensembling different models to increase performance: adaboost, bagging, random forest, gradient boost and stacking model. The outcome was a prediction of the status: cancelled, completed or active.

For data processing, they transformed the textual data by removing capitals and stop-words and by stemming the words. These steps reduce the required processing power, something valuable for the larger data set used in this study. The downside of removing stop-words, is that sequence modelling of the text is no longer possible. The textual data was then processed to be used as input for the model through NLP techniques. Bag of words was considered, but deemed too impractical for large data sets, which is also our case. The pre-trained text model BART was used in addition to Latent Dirichlet Allocation (LDA), which automates topic modelling and can use its output of a number of topics as input for the model.

Choetkiertikul et al.[2] looked at both individual issues and related issues through links, and then used collective inference to make predictions for related issues simultaniously. The individual issues done through the following features: discussion time, waiting time, tyep, task repetition, priority, changing of priority, no of fix versions, no of affect versions, changing of description, reporter reputation, developer's workload, percentage of delayed tasks, number of votes and number of watchers. The related issues are done in both an explicit and implicit manner. Explicitly, every

issue is linked to others and thus a relation is established. Implicitly, the authors created a model that creates its own links based on the individual issues' features.

3.3 Machine learning models

Todo before the next deadline on methodology

4 METHODOLOGY

4.1 Machine learning models for risk assessment

Based on the EDA and related work, this study will use the following methods and achieve the following goals:

* pre-process the textual data in the same manner as Meerman

* use ensembling as well, due to the variance and size of the input data

* make use of the NLP methods as done by Meerman, to reduce required processing power.

* our model will make a prediction on status (at the end of a sprint), number of changes of the field sprint, duedate, resolutiondate.

4.2 Methods to share risk control

Besides having a model that can assess the risk, it also needs to share which specific elements are causing the risk. These need to be reported to the team.

The team should be able to adjust the user stories added to a sprint and immediately receive feedback on the risk score of the current sprint planning.

Stretch ambition might be to automatically suggest which user stories from the top of the product backlog should be added to a sprint. For this to become a reality, team velocity based on story points should be present, possibly from other models.

A platform to offer such a service could be through Atlassian Marketplace, where various Jira Apps are offered to be used inside the planning software.

5 RISK ASSESSMENT

A few potential risks can be identified regarding the research as described in this thesis design. In this section, these risks will be discussed, and if needed, a back-up plan will be formulated.

While finding data sets including product backlogs with user stories is relatively simple, it will only be useful if it includes sprint data. This can be resolved by using data sets that have not yet been processed to remove this information.

The found data sets from different planning software might require different types of processing. Fortunately it is already labeled, so it should comprise mostly of cleaning the data sets. Another benefit is the availability of good scripting skills and available budget to outsource it.

Besides different models in which the data resides, entries in the data sets can widely vary between organisations and even between teams of the same organisation. This might introduce a risk of the model only being able to properly train on its own data set. This could possibly be turned around into an advantage, because the moment this occurs and is resolved, it should mean that the data model is applicable to new data sets as well.

For this study, I am dependent on previous study to supply the ML model. In the worst case scenario, I should be able to fall back to more basic models found in literature.

There is always the risk of the ML model accuracy not being significant. If this is the case, the main focus of the project should switch from risk assessment to risk control or vise versa.

6 PROJECT PLANNING

Figure 1 below shows an overview of the week-by-week planning for the thesis project. Yellow bars are the deadlines weeks set by UvA as feedback moments. Calendar week 4 is Monday 14 of February, week 26 is Monday 27 of June.

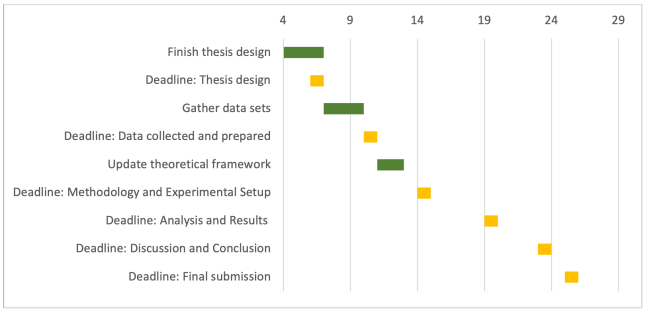


Figure 4: Gantt chart on planning per calendar week.

7 FUTURE WORK

Look at project level instead of jira-repo level.

REFERENCES

[1] [n.d.]. *Story level risk assessment in agile software development using machine learning.* Master's thesis.

[2] Morakot Choetkiertikul, Hoa Khanh Dam, Truyen Tran, and Aditya Ghose. 2017. Predicting the delay of issues with due dates in software projects. *Empirical Software Engineering* 22, 3 (2017), 1223–1263.

[3] David Cohen, Mikael Lindvall, and Patricia Costa. 2004. An introduction to agile methods. *Adv. Comput.* 62, 03 (2004), 1–66.

[4] Pete Deemer, Gabrielle Benefield, Craig Larman, and Bas Vodde. 2010. The scrum primer. *Scrum Primer is an in-depth introduction to the theory and practice of Scrum, albeit primarily from a software development perspective, available at: http://assets. scrumtraininginstitute. com/downloads/1/scrumprimer121. pdf* 1285931497 (2010), 15.

[5] Torgeir Dingsøyr and Yvan Petit. 2021. Managing layers of risk: Uncertainty in large development programs combining agile software development and traditional project management. *arXiv preprint arXiv:2103.09034* (2021).

[6] Philippe Kruchten, Robert L Nord, and Ipek Ozkaya. 2012. Technical debt: From metaphor to theory and practice. *Ieee software* 29, 6 (2012), 18–21.

[7] Lloyd Montgomery, Clara Lüders, and Walid Maalej. 2022. Jira: An Alternative Issue Tracking Dataset. *arXiv preprint arXiv:2201.08368* (2022).

[8] Lubna Siddique and Bassam A Hussein. 2014. Practical insight about risk management process in agile software projects in Norway. In *2014 IEEE International Technology Management Conference*. IEEE, 1–4.

[9] Wojciech Walczak and Dorota Kuchta. 2013. Risks characteristic to Agile project management methodologies and responses to them. *Operations Research and Decisions* 23 (2013).