



A Project Tracking Tool for Scrum Projects with Machine Learning Support for Cost Estimation

Kasi Periyasamy and Joshua Chianelli

University of Wisconsin-La Crosse, La Crosse, Wisconsin, U.S.A.
{kperiyasamy, chianell.joshua}@uwlax.edu

Abstract

Cost estimation in software development is very important because it not only gives an idea to all stakeholders on how long it takes to complete the product under development, but it also mandates tracking development activities so that the project does not overrun on time or budget. Several cost estimation models have been reported in the literature for software development using traditional life cycle models but there are only a few ad hoc methods for software projects that used agile methods. This paper describes the design and implementation of a project tracking tool for software projects that are developed using the agile method Scrum. The users of the tool can closely monitor the progress of user stories, sprint tasks and test cases inducted into a scrum board. The tool also supports cost estimation of the project based on user stories and sprint tasks. For every user story, the tool provides a measure of hardship to implement in terms of story points, and for every sprint task, it gives the anticipated completion time. The tool uses machine learning support for continuous monitoring of efforts based on sprint tasks. The effectiveness of the tool has been tested using three different graduate course projects.

1 Introduction

Recently, software developers switch to agile methods which are believed to improve the quality of software products being developed. A major difference between traditional software development methods (e.g., waterfall method) and agile methods is that the latter focus mainly on coding and testing phases, ignoring the need for extensive and structured documentation. Traditional methods use detailed requirements and design documents in order to thoroughly analyze the problem before coding. This approach more or less leads to frozen requirements. In contrary, agile methods prefer to be flexible in terms of changing the requirements at any time. Some developers use hybrid approaches in which they do develop some initial requirements and design documents, mainly to understand the problem domain before coding. However, these documents are not expected to be well-structured as required by traditional methods. *Scrum* is one of the popular agile methods used in industries[14]. Most developers tailor the scrum method to their needs but the general principles, as explained in [14] for a scrum method, are the same.

Cost estimation for software development is a crucial and daunting task in scrum. A lot of cost estimation models were reported in the literature[3, 15, 11], but many of these models

became obsolete because of changes in technology and development methods. Most of these models, including those that are based on use cases, predominantly use requirements specifications as the source for initial cost estimation. Consequently, they all rely on well-structured documents to specify the requirements. Since agile methods use mostly unstructured requirements specification (e.g., user stories used in scrum projects), it is difficult to use the cost estimation models published in the literature for agile methods. Besides, as evident from the agile manifesto [2], a welcoming aspect of any agile method is frequent changes in requirements. Because of these changing requirements, any initial cost estimation arrived at using the cost estimation models based on traditional methods would not be valid. Therefore, the cost estimation model for agile methods should be flexible enough to dynamically estimate the cost as and when requirements change.

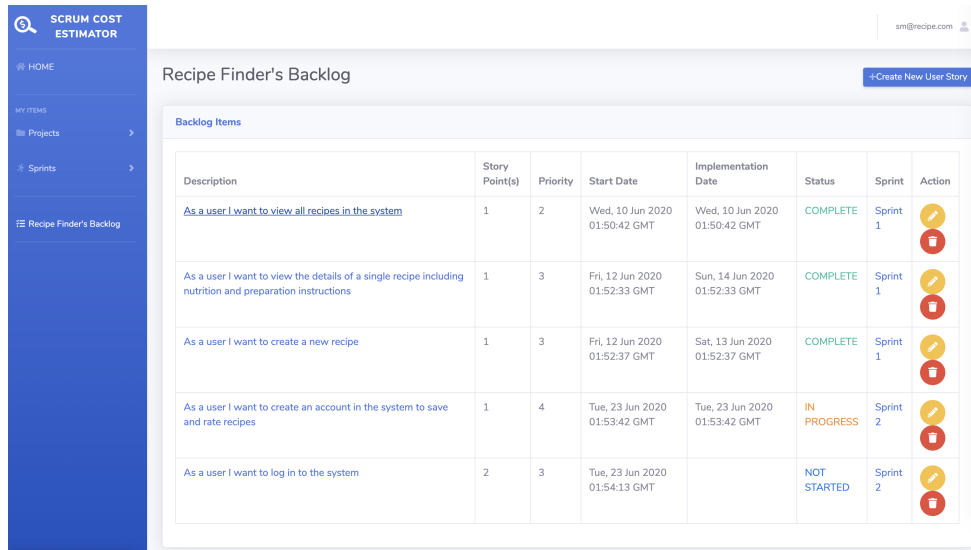
In this paper, we describe the design and implementation of a project tracking tool for scrum projects. The focus of the paper is two-fold: (i) to help software developers track the activities associated with a scrum project¹ and thereby viewing the progress of the project, and (ii) to estimate the cost of the overall project and constantly monitor to see whether the initial estimation stays as it is. If the estimate swings around (as expected in any software development project using agile methods), the tool will be able to indicate how far is the current estimate differs from the initial estimate. We used a machine learning approach so that the tool continuously updates its power of estimation.

Related Work

As mentioned earlier, most cost estimation models published in the literature fit well for traditional software development using waterfall and incremental prototyping models. Research on cost estimation models for agile software development is still in its infancy. Rosa and others [13] discuss the early phase cost estimation models for agile processes which focus more on getting an initial estimation based on the problem at hand. This approach is somewhat similar to the traditional models. Kang and others [8] used a model-based approach in which they develop an initial model which is dynamically updated as the development progresses. This approach seems to work better than Rosa and his team, but the work involves additional tasks to update the model and the cost estimation process. Popli and Chauhan [12] used a sprint-point based technique which is similar to use-case based approach used by other researchers [11]. Their approach is based on the sprint tasks and helps estimate the cost for each sprint separately. Some researchers [10, 9] believed that the cost estimation of an agile process continues to swing due to the technical debt that occurs during each sprint. Therefore, they focused on their cost estimation algorithms based on calculating the technical debt and thereby estimating the overall cost. Adnan and Afzal [1] used a scrum ontology model and description logic with multiagents to gather knowledge from various activities that occur in a scrum-based project and use this information for guiding the scrum master. Gandomani and others [7] discuss why the ‘Planning Poker’ approach (one of the commonly used techniques for effort estimation in agile methods) is not reliable. They assert that more concrete information is needed than consensus or averaging the size of user stories used as a source for initial cost estimation.

Our work reported in this paper uses a technique similar to the one used by Popli and Chauhan [12]. One of the authors this paper previously worked on cost estimation using use cases [11] and hence a similar approach using user stories and sprint tasks was believed to be feasible. So we enhanced Popli and Chauhan’s model with machine learning support.

¹We use the term ‘scrum project’ to refer to a software development project using the scrum agile method.



Description	Story Point(s)	Priority	Start Date	Implementation Date	Status	Sprint	Action
As a user I want to view all recipes in the system	1	2	Wed, 10 Jun 2020 01:50:42 GMT	Wed, 10 Jun 2020 01:50:42 GMT	COMPLETE	Sprint 1	✓ 🗑️
As a user I want to view the details of a single recipe including nutrition and preparation instructions	1	3	Fri, 12 Jun 2020 01:52:33 GMT	Sun, 14 Jun 2020 01:52:33 GMT	COMPLETE	Sprint 1	✓ 🗑️
As a user I want to create a new recipe	1	3	Fri, 12 Jun 2020 01:52:37 GMT	Sat, 13 Jun 2020 01:52:37 GMT	COMPLETE	Sprint 1	✓ 🗑️
As a user I want to create an account in the system to save and rate recipes	1	4	Tue, 23 Jun 2020 01:53:42 GMT	Tue, 23 Jun 2020 01:53:42 GMT	IN PROGRESS	Sprint 2	✓ 🗑️
As a user I want to log in to the system	2	3	Tue, 23 Jun 2020 01:54:13 GMT		NOT STARTED	Sprint 2	✓ 🗑️

Figure 1: Product Backlog describing user stories

2 Project Tracking Tool

The tool described in this paper provides extensive support for tracking the progress of a software development project that uses the scrum agile method. Even though there are a number of books and papers written on the scrum method, we would like to briefly outline the development process in scrum so that the readers will understand how the tool tracks the progress.

Scrum is an iterative development model in which the software product is incrementally developed. Each iteration is a fixed time-box and is called a *sprint*. The duration of a sprint is selected to be a shorter time period, usually two to three weeks. At the end of each sprint, a ‘potentially shippable’ product is expected to be delivered to the stakeholders. By ‘potentially shippable’, it means that the stakeholders should be able to use the partial product with limited features.

The requirements for the product are written in the form of user stories. A user story is a short description of one particular functionality that the product should include. These user stories are prioritized in each sprint and are kept in a list called ‘product backlog’. To support tracking of the progress of the project, the time-stamp (date and time) at which a user story is inducted into the product backlog is recorded. User stories can be added or changed at any time during the development, although they are normally changed at the beginning of each sprint. To describe a user story, we used the following notation which is a slightly modified version of the one given in [5]. A sample of user stories is shown in Figure 1. An obvious constraint in the product backlog is that ‘Start Date’ must be the same date or earlier one compared to ‘Implementation Date’ and must be within the sprint. The column ‘Story Point’ indicates a measure of anticipated efforts (time taken) to complete the associated user story; this will be discussed later in the paper.

During each sprint, a subset of user stories are selected for implementation from the product backlog by the development team. The time-stamp at which a user story is selected for implementation is also recorded. This time-stamp indicates the beginning of the life cycle for

As a user I want to view all recipes in the system's Sprint Tasks

SCRUM COST ESTIMATOR

sm@recipe.com

HOME

MY FILES

Projects

Sprints

Recipe Finder's Backlog

Create New Sprint Task

List of tasks

Title	Description	Priority	Start Date	Implementation Date	Complete Date	Status	Estimated Time (hours)	Actual Time	Assignee	Action
Create PostgreSQL database with recipe table	Recipe table should have attributes of name, ingredients, instructions, rating, and image	3	Wed, 10 Jun 2020 01:55:51 GMT	Thu, 11 Jun 2020 01:55:51 GMT	Thu, 11 Jun 2020 01:55:51 GMT	COMPLETE	2	2	dev@recipe.com	
Set up homepage backend controller	Create backend route for homepage that returns 200 status	2	Wed, 10 Jun 2020 01:56:36 GMT	Fri, 12 Jun 2020 01:56:36 GMT		COMPLETE	2	1	dev@recipe.com	

Copyright © Cost Estimator

Figure 2: Sprint Backlog describing sprint tasks associated with one user story

the user story. Each user story in the subset is divided into one or more sprint tasks. Generally, a sprint task is an activity that could be implemented by one developer in a day or two. Based on this convention, the development team could anticipate finishing the implementation of the selected user stories within the sprint. These sprint tasks are stored in a list called 'sprint backlog'. Like user stories, the time-stamp at which each sprint task is inducted into the sprint backlog is also recorded. Next, each developer in the team is assigned a subset of the sprint tasks to be completed within the current sprint with the goal that the integration of all the completed sprint tasks would lead to a 'potentially shippable' product. Once the sprint tasks are assigned, the developers start working on each assigned sprint task. The developer is required to record the time-stamp at which he/she starts the implementation of the sprint task. Like user stories, this time-stamp indicates the beginning of the life cycle of a sprint task. When the developer completes the sprint task, he/she is required to record the time-stamp at which the task is completed. This second time-stamp indicates the end of the sprint task's life cycle. The difference between the starting and ending time-stamps of a sprint task is recorded as the actual time taken to complete the sprint task. This metric is stored and used for further analysis. The sum of completion time of all sprint tasks associated with a user story represents the time taken for a user story. Currently, in the tool, this metric is not stored anywhere and can be computed at any time. The structure to describe the sprint tasks is shown in Figure 2. The 'Implementation Date' in sprint backlog refers to the date in which the sprint task was started for implementation. As in product backlog, there is a tight constraint among the three dates listed in the table - 'Start Date' must be the same or earlier than the 'Implementation Date' which must be the same or earlier than the 'Complete Date'.

For each sprint task, one or more test cases are generated to ensure that the sprint task is correctly implemented. These test cases may include both unit test cases and integration test cases. Unlike user stories and sprint tasks, test cases do not have a separate life cycle. Instead, the time spent on testing is included in the implementation of the sprint task. See Figure 3 which shows a set of test cases for one of the sprint tasks.

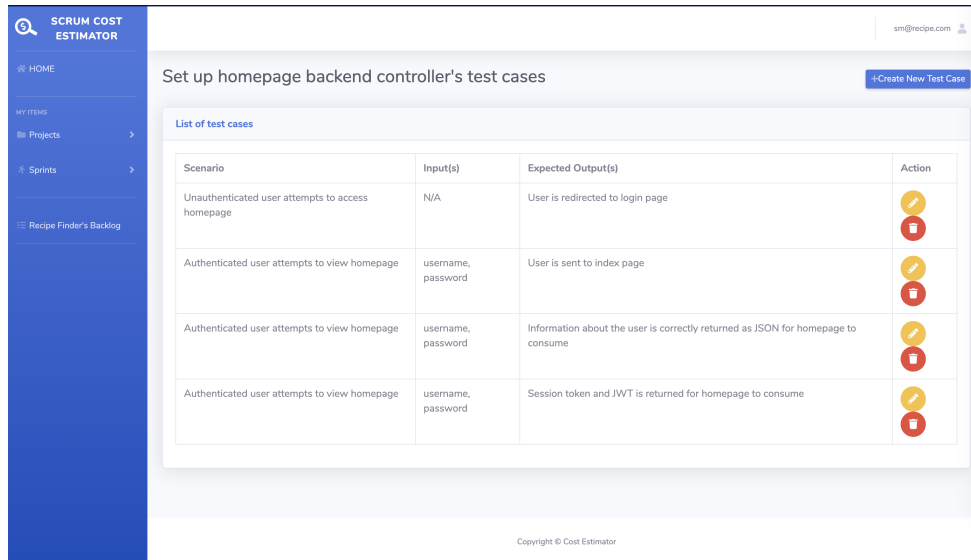


Figure 3: Test cases associated with one sprint task

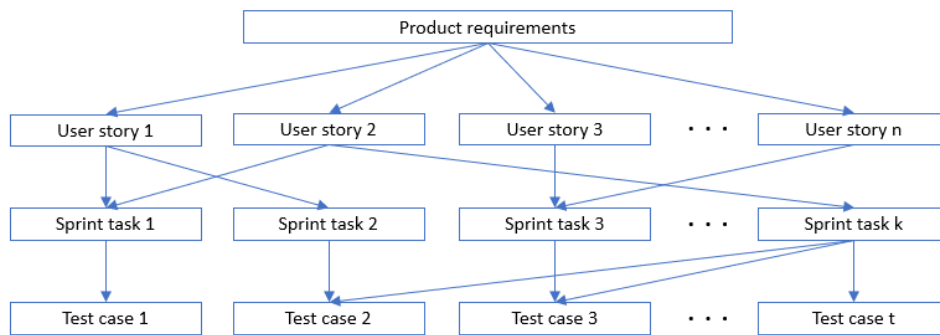


Figure 4: Relationships between user stories, sprint tasks and test cases

There is a many-to-many relationship between user stories and sprint tasks. Consider, for example, two user stories titled “login as an administrator” and “login as a scrum master”. The application may have different formatting requirements for the username of an administrator and that of the scrum master. Therefore, format validation of username may be implemented using two different sprint tasks. However, once the format is verified, the retrieval of the login entry from the login table in the database can be handled by one sprint task which may be shared by the two user stories. Similarly, there is a many-to-many relationship between sprint tasks and test cases. Figure 4 shows the relationships between user stories, sprint tasks and test cases.

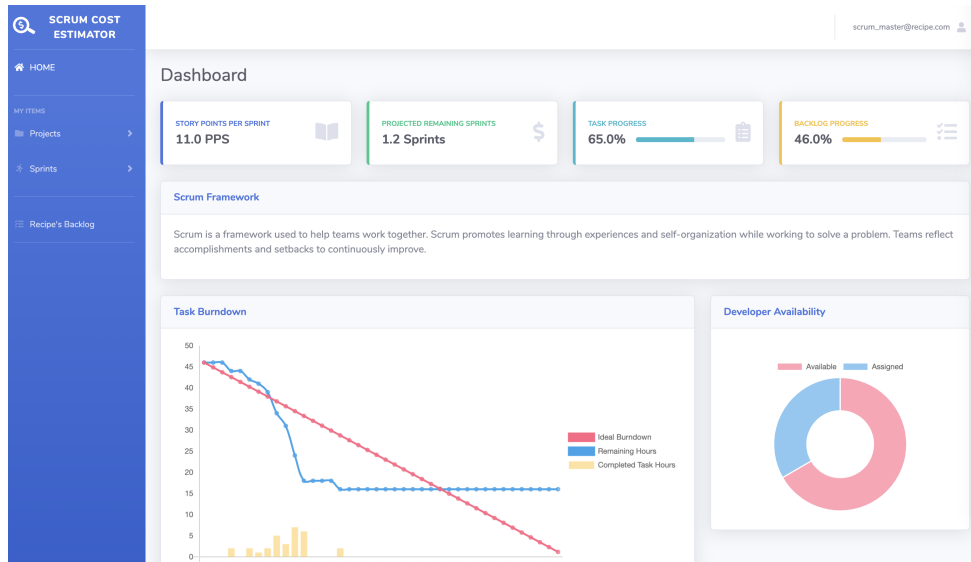


Figure 5: Dashboard view for a user in the development team

Initiating and tracking a project

The project tracking tool comes with an administrator account who creates and maintains other user accounts. The users participating in a scrum project are product owners, scrum master and developers. All users must have registered with the system and must have their accounts already set up by the administrator. Other than the administrator, only a scrum master can create a new project and assign developers to the project. Once a project is created, the scrum master will have full control of the project by entering and managing information about the project, and creating and maintaining the sprints. During each sprint,

- the scrum master and the product owner will have access rights to create new stories, as well as to update and to re-prioritize user stories;
- the scrum master and developers will have access rights to create and maintain sprint tasks for each user story; and
- the scrum master and developers will have access rights to create and maintain test cases for each sprint task.

Figure 5 shows the dashboard view when a user in the development team logs in. The dashboard displays a lot of information including the burndown chart of the current project, number of story points per sprint, remaining sprints yet to be completed, velocity and percentage of completion user stories from product backlog.

3 Cost Estimation

The tool comes with a support for cost estimation for a scrum project. It uses an internal storage called *Data Collector* which holds information on previously completed projects. This information includes user stories along with story points (to be discussed shortly), and sprint

tasks along with the time to complete each of those tasks. The sum of completion times of the sprint tasks associated with a user story will thus give the time taken to complete the user story. In a similar way, the sum of completion times of all user stories that were implemented will give the time taken to complete the project.

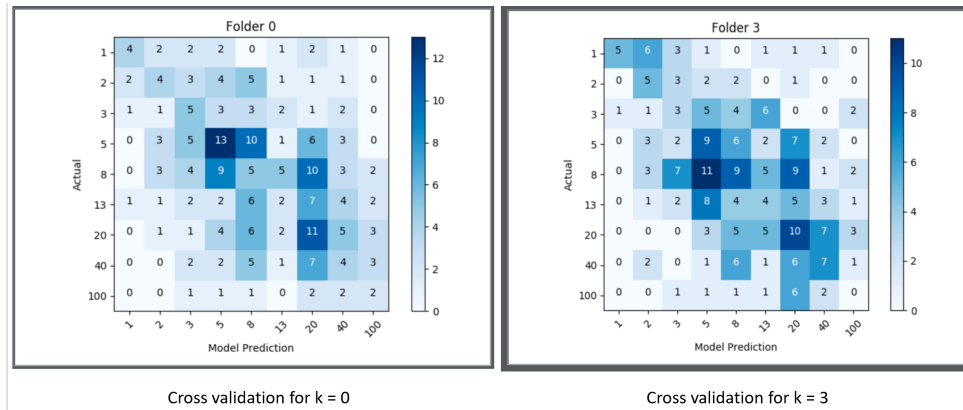
When a new user story is created for the current project, the tool compares the user story with those of the previous projects and gives a rough estimation of how hard or easy it is to implement the user story. This measure is given in terms of story points which are commonly used as a measure for specifying the size of a user story [6]. The range of values for story points in this project has been chosen as [1, 2, 3, 5, 8, 13, 20, 40, 100]. This is a slightly modified version of Fibonacci sequence. The reason for choosing these values was based on Weber's law which indicates that the difference between two situations of hardships can be better explained in terms of percentage. Accordingly, the difference between 1 and 2 is 100% while the difference between 20 and 21 is only 5%. So, when a user story is given a story point value in this range, it is easier for the developer to compare the complexity of implementing the user story, rather than giving a rough estimation of the number of hours to implement it (which is not easy to do when there is not much information about implementation given in the user story). The tool uses a machine learning library called *fastText* which applies a supervisory mode of training of a labeled data set. This data set includes user stories and their predefined story point values imported from several previous projects.

While searching for a user story, the tool uses *Elasticsearch* to query the stored user stories and displays closely matching options. The user has the choice of selecting a user story from one of the displayed options in which case the associated story point will be displayed. If the user wants to create a new entry instead, it will be added to the pool when the tool is retrained. Currently, the tool has data collected from 16 large open source projects containing more than 23,000 user stories, taken from [4]. These projects came from Apache, Atlassian, Moodle and others.

For sprint tasks, the tool gives estimation in terms of the number of hours because a sprint task has sufficient information on the amount of code to be written (just like a use case in UML). Moreover, in scrum projects, most sprint tasks are expected to be completed in a day or two. Similar to user stories, when a user enters a sprint task, the tool uses *Elasticsearch* to display closely matching sprint tasks from previous projects. If the user selects one of the matching sprint tasks, the user can see the actual completion time of the selected task which would give an estimation for the completion time of the current task.

The *Data Collector* can be used to import data from other projects. While importing new data, the tool verifies the compatibility of new data with the format currently used by the tool, and also trains itself in order to make sure that the new data is available for prediction immediately. Retraining is done in parallel to any project activity which is currently active and so it does not affect the progress of any active projects. Actions such as detecting missing values, removing stop words, removing duplicates, and removing incorrect identifiers (story point values that are not in the stored set) are all done before retraining. These actions together constitute 'data cleaning', an important component of a machine learning process.

To test the accuracy of the model, a k-fold cross validation was used on a set of over 1000 user stories from Moodle's project. The model was executed with a k-value of 5 and gave a testing accuracy of 24.21%. Figure 6 shows the confusion matrix for two k-values.

Figure 6: Cross validation of the model for $k = 0$ and $k = 3$

4 Conclusion and Future Work

Currently, agile methods are used for software development because of their two major advantages - requirements can be changed at any time, and no need for detailed and structured documentation needed before coding. Scrum is one of the popular agile methods used in many industries. This paper describes the design and implementation of a tool that supports tracking various activities in a scrum project, such as creating user stories, sprint tasks and test cases and also maintaining their relationships. In addition, the tool supports cost estimation of a scrum project based on its sprint tasks. The tool uses machine learning libraries to estimate the hardship of user stories and estimated completion times of sprint tasks by comparing their counterparts from previously completed projects. The tool has been tested with three graduate course projects.

Unlike cost estimation models applied to traditional software development projects, the tool does not give an initial estimation for the entire project. This is because the estimation for the entire project is evaluated as the sum of estimation of completion time of all sprint tasks. But sprint tasks are introduced only at the beginning of each sprint. Therefore, the cost estimation is a dynamic ongoing process. If the user likes to get an overall estimate for the entire project, the user can list all user stories at once and try to come up with all sprint tasks associated with those user stories. This will be equivalent to planning and designing the whole system before coding and hence will follow the traditional approach. In summary, there is no initial estimation of cost for the entire project in scrum.

While a new sprint task is inducted with an estimated completion time, a new user story is associated with a story point. The latter only describes the hardship of implementation by comparing with user stories from previous projects. But there is no direct connection between the story point of a user story and the estimated completion times of the sprint tasks associated with the user story. We would like to further explore and maintain the relationship between story points of user stories and estimated completion times of sprint tasks.

The tool uses user stories and sprint tasks from previous projects for estimation. The user has the option of reusing those user stories and sprint tasks. However, based on the publications on software reuse, it is not easy to find an exact match of a user story or a sprint task. For example, even if two software products for the same application, say for a bank or commerce application, are considered, there will be considerable differences between user stories of the

two products. Cost estimation models such as COCOMO and Function Point use cost drivers to fine tune an initial estimation. We would like to extend our tool by introducing cost drivers to fine tune the estimation of story points as well as the completion times of sprint tasks.

References

- [1] Muhammad Adnan and Muhammad Afzal. Ontology based multiagent effort estimation system for scrum agile method. *IEEE Access*, 5:25993–26005, November 2017.
- [2] Kent Beck, James Grenning, Robert C. Martin, Mike Beedle, Jim Highsmith, Steve Mellor, Arie van Bennekum, Andrew Hunt, Ken Schwaber, Alistair Cockburn, Ron Jeffries, Jeff Sutherland, Ward Cunningham, Jon Kern, Dave Thomas, Martin Fowler, and Brian Marick. Manifesto for agile software development. <http://agilemanifesto.org>, 2001.
- [3] Barry Boehm, Chris Abts, Winsor Brown, Sunita Chulani, Bradford K. Clark, Ellis Horowitz, Ray Madachy, Donald J. Reifer, and Bert Steece. *Software Cost Estimation with COCOMO-II*. Prentice Hall, 2000.
- [4] Morakot Choetkiertikul, Hoa Khanh Dam, Truyen Tran, Trang Pham, Adhitya Ghose, and Tim Menzies. A deep learning model for estimating story points. *IEEE Transactions on Software Engineering*, 45(7):637–656, 2018.
- [5] Mike Cohn. Advantages of the ‘as a user, i want’ the template. <https://www.mountangoatsoftware.com/blog/advantages-of-the-as-a-user-i-want-user-story-template>.
- [6] Mike Cohn. *Agile Estimating and Planning*. Pearson Education, 2005.
- [7] Taghi Javdani Gandomani, Hamidreza Faraji, and Mahsa Radnejad. Planning poker in cost estimation in agile methods: Averaging vs consensus. In *IEEE 5th International Conference on Knowledge-based Engineering and Innovation (KBEI)*, pages 66–71, February 2019.
- [8] Sungjoo Kang, Okjoo Choi, and Jongmoon Baik. Model-based dynamic cost estimation and tracking method for agile software development. In *Proceedings of the IEEE/ACIS 9th International Conference on Computer and Information Science*, pages 743–748. IEEE, August 2010.
- [9] Antonio Martini and Jan Bosch. The magnificent seven: Towards a systematic estimation of technical debt interest. In *Proceedings of the XP’17 Workshops*, May 22–26 2017.
- [10] Ariadi Nugroho, Joost Visser, and Tobias Kuipers. An empirical model of technical debt and interest. In *Proceedings of the 2nd Workshop on Managing Technical Debt, part of International Conference on Software Engineering (ICSE 2011)*, pages 1–8, May 21–28 2011.
- [11] Kasi Periyasamy and Aditi Ghode. Cost estimation using extended use case points (e-ucp). In *Proceedings of the International Conference on Computational Intelligence and Software Engineering (CiSE’09)*, pages 2556–2560, Wuhan, China, December 11–13 2009.
- [12] Rashmi Popli and Naresh Chauhan. A sprint-point based estimation technique in scrum. In *Proceedings of the International Conference on Information Systems and Computer Networks (ISCON’13)*, pages 98–103. IEEE, March 9–10 2013.
- [13] Wilson Rosa, Raymond Madachy, Bradford Clark, and Barry Boehm. Early phase cost models for agile software processes in us dod. In *IEEE/ACM International Symposium on Empirical Software Engineering and Measurement*, pages 30–37, November 19–20 2017.
- [14] Kenneth S Rubin. *Essential Scrum: A Practical Guide to the most popular Agile Process*. Addison-Wesley Professional, 2012.
- [15] Adam Trendowicz and Ross Jeffery. *Software Project Effort Estimation: Foundation and Best Practices Guidelines for Success*. Springer, 2014.