Author/s:
Pasuksmit, J;Thongtanunam, P;Karunasekera, S

Title:
Story points changes in agile iterative development: An empirical study and a prediction approach

Date:
2022-11-01

Citation:
Pasuksmit, J., Thongtanunam, P. & Karunasekera, S. (2022). Story points changes in agile iterative development: An empirical study and a prediction approach. Empirical Software Engineering, 27 (6), https://doi.org/10.1007/s10664-022-10192-9.

Persistent Link:
https://hdl.handle.net/11343/320187

Empirical Software Engineering (2022) 27: 156
https://doi.org/10.1007/s10664-022-10192-9

Story points changes in agile iterative development

An empirical study and a prediction approach

Jirat Pasuksmit[1] · **Patanamon Thongtanunam**[1] · **Shanika Karunasekera**[1]

Accepted: 13 June 2022 / Published online: 10 August 2022

Abstract

Story Points (SP) are an effort unit that is used to represent the relative effort of a work item. In Agile software development, SP allows a development team to estimate their delivery capacity and facilitate the sprint planning activities. Although Agile embraces changes, SP changes after the sprint planning may negatively impact the sprint plan. To minimize the impact, there is a need to better understand the SP changes and an automated approach to predict the SP changes. Hence, to better understand the SP changes, we examine the prevalence, accuracy, and impact of information changes on SP changes. Through the analyses based on 19,349 work items spread across seven open-source projects, we find that on average, 10% of the work items have SP changes. These work items typically have SP value increased by 58%-100% relative to the initial SP value when they were assigned to a sprint. We also find that the unchanged SP reflect the development time better than the changed SP. Our qualitative analysis shows that the work items with changed SP often have the information changes relating to updating the scope of work. Our empirical results suggest that SP and the scope of work should be reviewed prior or during sprint planning to achieve a reliable sprint plan. Yet, it could be a tedious task to review all work items in the product (or sprint) backlog. Therefore, we develop a classifier to predict whether a work item will have SP changes after being assigned to a sprint. Our classifier achieves an AUC of 0.69-0.8, which is significantly better than the baselines. Our results suggest that to better manage and prepare for the unreliability in SP estimation, the team can leverage our insights and the classifier during the sprint planning. To facilitate future studies, we provide the replication package and the datasets, which are available online.

Communicated by: Andrian Marcus

✉ Jirat Pasuksmit
jpasuksmit@student.unimelb.edu.au; jiratpasuksmit@gmail.com

Patanamon Thongtanunam
patanamon.t@unimelb.edu.au

Shanika Karunasekera
karus@unimelb.edu.au

[1] School of Computing and Information Systems, The University of Melbourne, Melbourne, Australia

Keywords Software effort estimation · Story points · Mining software repositories

1 Introduction

Story Points (SP) are a relative effort unit that is commonly used to represent the effort of a work item in Agile software development process (Usman and Britto 2016). SP are used to facilitate the planning activities of a working iteration (e.g., 'sprint' in Scrum). The team mainly uses SP to allocate work items for a sprint. SP are also used to measure the team's delivery capacity, i.e., the available capacity of the team to work in the upcoming sprint, which helps the team plan the sprint in order to deliver a product increment (Coelho and Basu 2012). (Cohn 2006)[p.7] also suggested that "reliable estimates enable reliable delivery."

Typically, SP of a work item are initially estimated during product backlog refinement (a.k.a. backlog grooming) (Fowler 2019). Ideally, accurate SP should reflect the development time (Cohn 2006). Thus, the team can achieve a reliable sprint plan, i.e., the selected work items are completed within the sprint. As Agile methodologies are amenable to accepting and adapting to changes, the initially estimated SP may not reflect the relative size of the work item due to changes that have occurred. Therefore, prior to finalizing the sprint plan, the team may revisit the SP to acquire confidence that the selected work items can be delivered within the sprint (Rubin 2012).

SP can be changed (re-estimated) when the relative size of the work item is changed (Cohn 2006). Intuitively, a change of SP indicates that the work item may be larger (or smaller) than initially estimated and the original SP value was inaccurate (i.e., not reflect the development time). Therefore, the change of SP after sprint planning may cause the sprint plan, which is created based on the SP, to become unreliable (Cohn 2006). The impact can be larger if the plan has been committed to the customers. Consecutive SP changes may cause the customer to be reluctant to make important decisions as they may presume that the estimation and the plan are unreliable (Cohn 2006). As suggested by (Prikladnicki et al. 2019), there is a need for an approach or tools to predict the possible unreliability of the estimates.

Hence, we first conduct a mixed-methods empirical study to understand the various aspects of SP changes (i.e., prevalence, accuracy, and the impact of information changes on SP changes) after the work items were assigned to a sprint. Then, to help the team better cope with the unreliability in SP estimation, our goal is to develop a classifier to predict the future SP changes of a work item. In this work, we mine the information that is recorded in the summary, description, and other related fields of a work item. Then, we develop classifiers using machine learning techniques to predict whether a work item will have SP changes after it is assigned to a sprint. We conduct a case study on seven open-source projects which are known to actively use SP and an online task management system (i.e., JIRA) to record the information of a work item. Based on 19,349 work items across these six projects, we answer the following research questions:

(RQ1) To what degree do Story Points change?

Motivation: SP may be changed to better reflect the relative effort of a work item (Cohn 2006). Ideally, SP should not be changed after it was assigned to a sprint (i.e., after the sprint was planned). Such SP changes may have a negative impact on the team, e.g., additional effort may be required to re-plan the sprint, the sprint plan may become unreliable, or the team may lose the customers' trust (Rubin 2012; Cohn 2006). Nevertheless, the team might

have a policy or norm that allows SP to be changed after the sprint plan is finalized (Hoda and Murugesan 2016; Masood et al. 2022; Pasuksmit et al. 2021). Yet, it is unknown whether SP changes after sprint planning are common and their sizes are significant or not. The high frequency of SP changes with a large changing size would suggest that the team may often encounter unreliable sprint plans. In this RQ, we quantitatively analyze the nature of SP changes (i.e., prevalence, changing size, and frequency) to shed light on whether SP change is a significant issue that often occurs or not.

Key findings: On average, only 10% of the studied work items have SP changes after they were assigned to a sprint, where 88% of them have SP changed only once. Moreover, an average of 57% of these work items have their SP increased by a typical size change of 58%-100% relative to the SP values when they were assigned to the sprint. These results suggest that even though SP are not frequently changed after sprint planning, the SP tend to be increased by a relatively large size change.

(RQ2) Do changed Story Points better reflect the development time than the ones that remain unchanged?

Motivation: To achieve a reliable sprint plan, SP should reflect the development time, i.e., the time spent to complete a work item (Cohn 2006). However, it is still unclear whether the SP that remain unchanged **after** sprint planning (i.e., unchanged SP) or the SP that are changed even **after** sprint planning (i.e., changed SP) better reflect the development time. Using the SP values that do not reflect the development time for sprint planning would lead the sprint plan to be unreliable. If the estimated SP do not reflect the development time, the allocated time for the sprint may not be sufficient. On the other hand, the estimated SP that can reflect the development time would allow the team to allocate the right amount of work to fit in the time box of the sprint. Hence, we perform a statistical analysis to empirically examine the relationship between the SP values and the development time.

Key findings: The unchanged SP share a stronger relationship with the development time than the changed SP (either the original SP values or their last values). This result indicates that the stable (unchanged) SP can better reflect the effort of a work item than the unstable (changed) SP. We also found that the last value of changed SP tends to share a stronger relationship with development time than the changed SP value when the work item was assigned to a sprint. The findings of this RQ highlight that creating a sprint plan based on unstable SP (i.e., the SP that will be changed after sprint planning) may prone to be an unreliable plan.

(RQ3) To what extent are Story Points changes are associated with information changes?

Motivation: Grounded Theory studies pointed out that the practitioners may re-estimate the SP when the information is changed (Hoda and Murugesan 2016; Bick et al. 2018; Masood et al. 2022). However, it is unclear whether this association is significant. Moreover, there are various possible types of information changes, e.g., scope changes or clarifications (Madampe et al. 2020; Rubin 2012). However, little is known what types of information changes could lead to SP changes. The findings of this analysis should help the practitioners save effort in maintaining SP by focusing on particular types of information changes that could lead to SP changes. Therefore, in this RQ, we first statistically analyze the association between the frequency of SP changes and information changes. Then, we further conduct a manual analysis to identify the types of information changes that were made to the work items that have resulted in an SP change.

Key findings: After the work items were assigned to a sprint, 33%-60% of the work items with changed SP have information changes, while only 9%-17% of the work items

with unchanged SP have information changes. The odds that a work item with changed SP will have information changes is 2.25-4.35 times higher than that of a work item with unchanged SP. From our manual analysis, 7%-41% of the work items with changed SP have information changes for updating the scope, while 60%-81% of the work items with unchanged SP have information changes for clarification. These results suggest that SP changes often occur along with information changes after the sprint planning. Moreover, the information changes for updating scope often occur in the work items with SP changes.

(RQ4) Can we predict whether a work item will have Story Points changes?

Motivation: SP changes indicate that the effort required for the work item becomes larger (or smaller) than expected. Indeed, our RQ1 shows that SP tend to be increased by a relatively large size change after a work item was assigned to a sprint. Moreover, our RQ2 shows that the SP that were changed after sprint planning do not reflect the development time better than the unchanged SP. However, it could be a tedious task for the team to revisit SP of all work items in the sprint backlog to improve the stability of the sprint plan. Therefore, we develop a classifier to predict whether a work item will likely to have SP changes after being assigned to a sprint. With our classifier, the team can save effort in sprint planning by focusing on revisiting a subset of work items that are predicted as having SP changes.

Key findings: Our classifier can predict whether a work item will have SP changes after it was assigned to a sprint. In particular, our LR classifier achieve an average AUC of 0.69-0.8, an average Balanced Accuracy of 0.6-0.68, and an average D2H of 0.33-0.47, which are better than the baselines. We find that *reporter-stable-sp-rate* and *text-gaussian* are the most influential metrics in our classifiers. These results suggest that our classifier can predict whether the SP will be changed after the work item is assigned to a sprint based on the available information. The past tendency of the work items reporter (*reporter-stable-sp-rate*) and likelihood estimated based on textual content (*text-gaussian*) are the most important characteristics to anticipate the stability of SP.

Key Contributions. This paper is the first to present: (1) An empirical result on the prevalence, frequency, and size of SP changes; (2) The impact that the estimation unreliability can have on the SP accuracy; (3) A manual categorization of the types of information changes that are associated with SP changes; (4) The classifier that can predict whether the SP will be changed based on the available information when a work item is assigned to a sprint. Our results suggest that the team can leverage our insights of SP changes and the classifier to better manage and prepare for the unreliability during the sprint. Additionally, we provide the replication package to facilitate the replication of our work.[1]

2 Background

In this section, we describe an ideal Agile iterative development and effort estimation in Agile. We also discuss the potential impact of SP changes based on the literature.

2.1 Agile Iterative Development

In Agile, a software development team plans, develops, and delivers a product increment in a time-boxed iteration (i.e., sprint). Figure 1 shows a typical Agile iterative development process (e.g., Scrum), which includes four main steps, i.e., (1) product backlog refinement,

[1] https://github.com/awsm-research/storypoints-changes

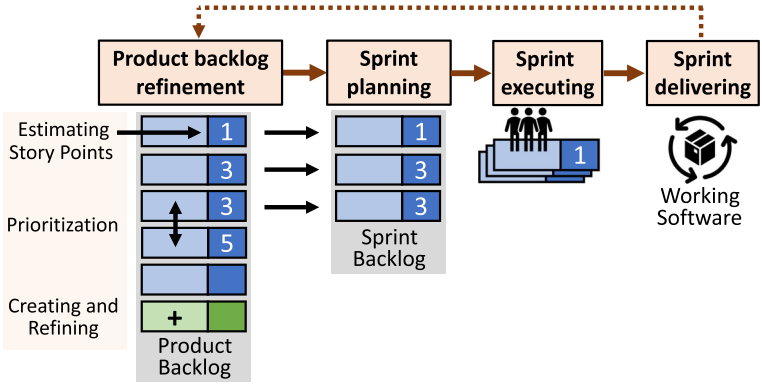


Fig. 1 A typical Agile iterative process (Scrum) (Dam et al. 2019)

(2) sprint planning, (3) sprint executing, and (4) sprint delivering. Product backlog refinement is the step that the team reviews the work items in the **product backlog**, i.e., a list of work items to develop the software (Sedano et al. 2019). Some work items in the product backlog may be large, i.e., Epics, which describe a high-level overview of a feature. During the product backlog refinement, the team refines Epics by creating a set of smaller work items (e.g., stories or tasks). Then, the team estimates the effort required to complete each work item and prioritize them.

At the beginning of each sprint, the team performs sprint planning to select the work items that the team will commit to delivering as a product increment. To do so, they first define the sprint goal and determine the team's **delivery capacity**, i.e., the available capacity of the team members to work in the sprint (Fowler 2019). Then, the team reviews and selects the work items that are aligned with the sprint goal and have estimated effort fit into the team's delivery capacity (Fowler 2019; Masood et al. 2022). The selected work items are then moved from the product backlog to the **sprint backlog**. These work items in the sprint backlog will be committed to be delivered to the customer within the sprint. Ideally, these work items should remain unchanged throughout the sprint (Fowler 2019; Rubin 2012). After the sprint planning, the team executes the sprint, i.e., implementing the work items in the sprint backlog. Finally, they deliver the implemented work items as a product increment to the customer when the sprint ends.

2.2 Effort Estimation in Agile

Effort estimation helps the team in sprint planning. When selecting work items for a sprint, the estimated effort of a work item is considered to ensure that the workload will fit the team's delivery capacity (Schwaber and Sutherland 2020). Additionally, the team's delivery capacity is also derived from the estimated effort. More specifically, the delivery capacity is determined based on the **velocity**, i.e., the total estimated effort of work items that the team delivered per sprint in the past (Fowler 2019). In addition, the velocity can be used to roughly predict when a software (or a release) should be delivered. For example, a team plans to release a software product that contains work items with a total effort of 100 units. If the team performs a two-weeks sprint with a velocity of 50, the team will require two

sprints (or four weeks) to deliver the software. Hence, to achieve a reliable software delivery, the estimated effort should be reliable (Cohn 2006).

Story Points (SP) are an effort unit that is commonly used in Agile (Usman and Britto 2016). SP reflects the relative effort of a work item (Fowler 2019). For instance, a work item that is assigned with two SP should take twice as much effort as a work item that is assigned with one SP (McConnell 2006). Typically, SP are estimated by the team using subjective decision methods based on team consensus, e.g., Planning Poker (Grenning 2002; Usman and Britto 2016). The team estimates SP of a work item by considering different factors based on the available information, e.g., complexity and size of the work item, and the development team's experiences (Rubin 2012; Usman and Britto 2016). Henceforth, we will use the term 'SP' interchangeably with the term 'estimated effort'.

2.3 Story Point Changes

SP are allowed to be changed (re-estimated) (Cohn 2006). During product backlog refinement or sprint planning, the team refines work items when detailed information is retrieved. If the refinement causes the SP to become inaccurate, the team may re-estimate the SP to better reflect the relative size of the refined work item (Bick et al. 2018). Then, the team will notify all stakeholders and adjust their plan if needed.

Nevertheless, given that the sprint plan is based on the estimated SP at the sprint planning, any changes to the SP after the sprint planning may impact the sprint plan. Intuitively, SP changes indicate that the effort required to implement a work item is larger (or smaller) than expected. Especially, if the SP are increased after the sprint planning, SP changes can negatively impact the sprint plan, i.e., the actual effort required may exceed the delivery capacity (Rubin 2012). Even though the sprint has not started yet, the team may still waste their effort in re-planning to accommodate the changes. The impact may be larger if the sprint plan is already be committed to the customer (Cohn 2006) or the sprint execution is already commenced (Rubin 2012). Some work items may be forced to complete within the sprint despite the increased SP (Hoda and Murugesan 2016). Yet, such a practice may negatively impact the quality of code and software (Cohn 2006). Furthermore, if several sprint deliveries were unsuccessful or partly successful (i.e., some work items were postponed) due to unreliable sprint plans and/or SP estimates, the team may not be able to build trust with the customer (Rubin 2012).

Achieving reliable SP estimation benefits the team in sprint planning (Cohn 2006). Reliable SP estimation leads to sustainable team workload allocation and enables reliable delivery that builds trust between the team and customer. However, in practice, SP are often initially estimated based on incomplete information. Such information may be discovered late or changed (Hoda and Murugesan 2016; Svensson et al. 2019), which may lead to the SP changes. To minimize the impact, there are needs to better understand the SP changes and an automated approach to predict the SP changes. Hence, in this work, we first conduct an empirical study to better understand SP changes in various aspects, i.e., prevalence, accuracy, the impact of information changes on SP changes. Then, we aim to develop an approach to predict whether the SP will be changed based on the available information when a work item is assigned to a sprint. The key motivation behind is that the changes are inevitable in Agile and forecasting the changes allows the teams to better prepare for the unreliability (Prikladnicki et al. 2019).

3 Case Study Design

This section presents our motivation for our research questions, our project selection, and data preparation approaches.

3.1 Studied Projects

In recent practice, the teams use an online task management system (e.g., JIRA) to record the work items in their product backlog (Hoda and Murugesan 2016; Choetkiertikul et al. 2019). To study the SP changes, we select the open-source projects that satisfy the following criteria.

Criterion 1 – Traceability: Since we will perform data analysis to empirically examine and predict the SP changes (i.e., the activity of SP assignments), we need the historical record to determine whether the SP were changed. Hence, we opt to study the projects where the activities are recorded in a task management system.

Criterion 2 – Active Story Points Usage: Since this study focuses on SP changes, we select the projects that actively use SP as a unit to represent the effort required to complete a work item. Based on this criterion, we select the projects that have a relatively large number of work items with SP as the projects with only few work items with SP may not actively use SP.

Criterion 3 – Story Points change norm or policy: Since we want to study the SP change, we need to ensure that SP changes are generally allowed in the project and the SP changes that we could observed were not occurred by mistakes. Hence, we opt to study the projects where their practice generally allows the SP to be changed by the norm or policy.

To satisfy our first criterion, we obtain a list of 16 Agile software projects provided by (Choetkiertikul et al. 2019), i.e., the state of the art effort estimation approach. These 16 projects use JIRA[2] to track their development tasks and SP assignment activities. Unfortunately, two of these 16 projects (i.e., JIRA Software and Mule Studio) no longer provide public access. Hence, 14 Agile software projects are remaining.

For our second criterion, we determine if a project has a relatively large proportion of work items with SP by examining the distributions of the proportion of work items across all the 14 studied projects. We observe that the proportions of work items with SP of 10 projects skewed towards greater than 10% (i.e., above the 25^{th} percentile). On the other hand, the proportions of work items with SP of the other four projects (AP, BB, MD, and TE) are relatively lower than the others (i.e., only 2% - 7%, which are under the 25^{th} percentile). This observation suggests that these four projects may not use SP as actively as other studied projects. Hence, we exclude these four projects from our study.

To satisfy our third criterion, we contact the core contributors of the remaining 10 projects (i.e., the developers who regularly contributed to the projects) to confirm whether their projects generally allow SP to be changed or not. We opt to contact the core contributors because they should be more involved in the effort estimation and sprint planning of the projects than the casual contributors. Therefore, we first identify the core contributors and their contact details from the official website of the projects. If their emails are not publicly available, we ask the contact person listed in the official websites or ask developers in the Slack channel of the project about the core contributors' emails. Then, we send an email to the core contributors to ask (1) whether their projects generally allow SP to be changed. In

[2] https://atlassian.com/software/jira

Table 1 An overview of the studied projects

Project	Period	#work items			
		Total	With SP (%)	Studied (%)	
Appcelerator Studio (AS)	03/2011-03/2019	5,925	2,724 (46%)	636 (23%)	
Data Management (DM)	03/2014-04/2019	18,678	11,480 (61%)	4,803 (42%)	
Apache Mesosphere (ME)	02/2011-04/2019	9,695	2,362 (24%)	1,678 (71%)	
Mule (MU)	03/2004-04/2019	15,640	4,890 (31%)	2,924 (60%)	
Titanium SDK/CLI (TI)	03/2011-04/2019	21,189	3,127 (15%)	1,630 (52%)	
Usergrid (UG)	11/2013-04/2019	1,338	337 (25%)	201 (60%)	
Spring XD (XD)	04/2013-11/2018	3,710	2,472 (67%)	2,030 (82%)	
Aptana Studio (AP)†	12/2003-02/2018	8,135	539 (7%)	n/a	
Bamboo (BB)†	04/2006-04/2019	13,602	465 (3%)	n/a	
Moodle (MD)†	04/2002-04/2019	62,316	1,009 (2%)	n/a	
Talend ESB (TE)†	12/2010-04/2019	15,623	821 (5%)	n/a	
Clover (CV)†	08/2007-01/2017	1,501	229 (15%)	n/a	
Duracloud (DC)†	11/2009-04/2019	1,085	572 (53%)	n/a	
Talend Data Quality (TD)†	02/2008-04/2019	14,222	1,524 (11%)	n/a	

*The projects above the double lines satisfy our selection criteria.

† The projects were excluded by the second criterion.

‡ The projects were excluded by the third criterion.

addition, to confirm that the projects use SP according to our Criterion 2, we also ask (2) do they estimate SP and when, (3) when do they assign the work items to the sprint, and (4) how long is the sprint length. We send 47 emails to the core contributors of the 10 projects and receive 12 responses (26% response rate). In summary, the core contributors of seven projects (i.e., AS, DM, ME, MU, TI, UG, and XD) confirm that SP are generally allowed to be changed in their projects. The core contributors of the other three projects (i.e., CV, DC, and TD) inform us that their projects generally do not allow the SP to be changed. Hence, we exclude these three projects from our study. Noted that we provided the detail of the core contributors' responses in the Section 4.

Finally, seven projects satisfied the three criteria. Table 1 shows an overview of the seven projects that we will use in this study. These projects are open-source projects. The software are developed and maintained by a team of developers or an organization with multiple teams. Appcelerator Studio (AS) is an Integrated Development Environment (IDE) tool for web and mobile application development.[3] Data Management (DM) is data management systems for storing and manipulating the data from Rubin Observatory.[4] Apache Mesosphere (ME) is a resource management and scheduling system for clusters.[5]

[3]https://docs.axway.com/bundle/Appcelerator_Studio_allOS_en/page/axway_appcelerator_studio.html

[4]https://www.lsst.org/about/dm

[5]Apache Mesosphere (https://mesos.apache.org), developed by the contributors from D2iQ (https://d2iq.com/about).

Mule (MU) is an integration platform that handles communication between subsystems.[6] Titanium SDK/CLI (TI) is a Software Development Kit (SDK) that used to develop multi-platforms applications.[7] Usergrid (UG) is a Backend-as-a-Service (BaaS) to enable a rapid development of web and mobile applications.[8] Spring XD (XD) is a system used for data ingestion, real-time analytics, batch processing, and data export to ease big data application development.[9]

3.2 Data Preparation

Our studied projects use JIRA to record their work items. A JIRA work item contains a summary, description, and other properties, including the estimated effort (i.e., SP) and sprint number (see Fig. 2). JIRA also has an activity log that records a history of changes in a work item, e.g., SP, summary, description changes. For example, Fig. 2 shows an activity log where the SP were changed from 3 to 5 along with the description of the work item.

To perform our empirical study, we mine the data recorded in JIRA. Below, we describe our data preparation approach, which consists of four main steps.

(DP-1) Collecting data. We download work items that are recorded in the JIRA repositories of the studied projects. We use the REST API provided by JIRA to query the work items that have SP. In this study, we will only focus on the work items that are closed to ensure that the work has already finished and the latest recorded SP are the final ones. In particular, we only collect the work items that have a status of *done*, *closed*, or *resolved*, and a resolution of *done*, *complete*, or *fixed*. Finally, we download the summary, description, other properties, and activity log.

(DP-2) Identifying the type of work item. Since each project has a different set of *types of work items*, we will only study the types that are commonly used among the studied projects, i.e., *Bug*, *Improvement*, *New Feature*, *Story*, and *Task*. We find that Mule is the only project that has *Enhancement Request*. We manually examine these work items and find that *Enhancement Request* can be considered as *Improvement*. Note that we exclude *Epic* work items out from our study since they are typically large, vague, and the team does not work on them (Rubin 2012).

(DP-3) Identifying Story Points changes. In this work, we focus only SP changes occur after sprint planning. However, since the time point of sprint planning was not recorded in JIRA, we use the time when the work item was assigned to a sprint as the reference time point for sprint planning. We believe that this time point can be represented as the time point of the sprint planning since it is suggested that the work items should be assigned to the sprint during sprint planning (Schwaber and Sutherland 2020). Moreover, the core contributors that we contacted also inform us that the work items are generally assigned to the sprint during sprint planning (see Section 3.1).

To identify whether a work item has SP change after sprint planning, we extract the SP values that were assigned to the work item from the activity log. If there is more than one unique SP values after the work item was assigned to a sprint, the work item is identified as a *work item with changed SP*. Otherwise, the work item is identified as a *work item with unchanged SP*. It is possible that developers may perform documentation (e.g., document

[6] https://github.com/mulesoft/mule

[7] https://appcelerator.com/titanium/titanium-sdk/4

[8] https://usergrid.apache.org/

[9] https://projects.spring.io/spring-xd

Fig. 2 An example of an activity log of a work item recorded in JIRA

the finalized SP values in JIRA) after sprint planning. Hence, we consider the one hour gap after the work item was assigned to a sprint as the documentation period.[10] Therefore, a work item is considered as having SP change if its SP were changed later than one hour after the work item was assigned to a sprint. Moreover, for the changed SP, we also identify the SP value at the sprint assignment time (i.e., changed SP_{sprint}) and the last SP value (i.e., changed SP_{last}).

Figure 3 shows three examples of our approach that identifies the changed SP. Work items #1 and #2 are identified as the *work items with unchanged SP* because they do not have any SP assignment event after they were assigned to a sprint. Work item #3 is identified as the *work item with changed SP* because it has an SP assignment event later than one hour after it was assigned to a sprint. In addition for Example #3, the SP value of 1 is identified as the changed SP_{sprint} and the SP value of 3 is identified as the changed SP_{last}.

(DP-4) Cleaning data. In this work, we aim to study the work items that the team worked on them and their SP were estimated before or during sprint planning. Therefore, we first exclude the work items that are assigned with too large (SP>100) or temporary (SP=0) SP values as the teams are not likely to work on them (Cohn 2006; Choetkiertikul et al. 2019). Then, we exclude the work items that have initial SP assigned after they were assigned to a sprint as these late-assigned SP values may not be used for the sprint plan. As justified in DP-3, the first hour after sprint planning could be the documentation period.

[10] We discuss this threshold in Section 6.3.

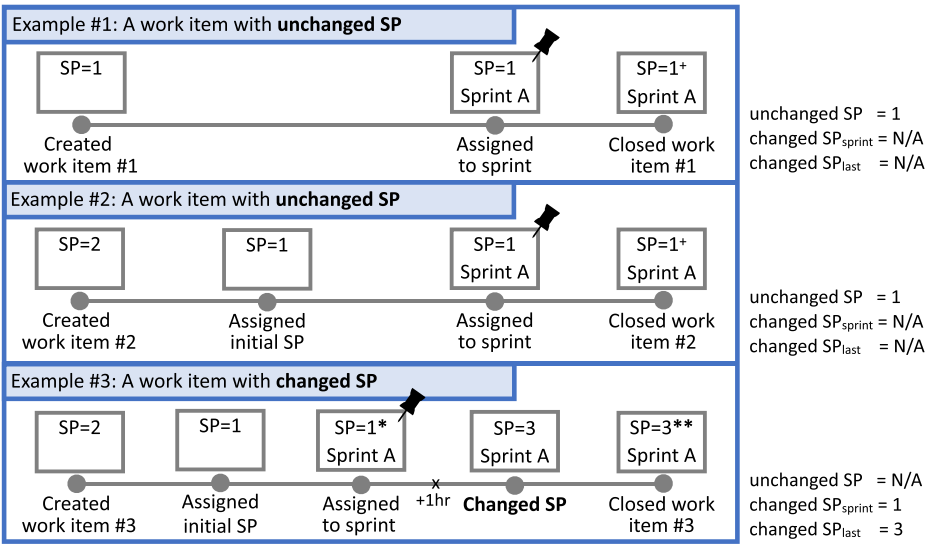


Fig. 3 Examples of the identification of SP changes based on the activity log of a work item. + : unchanged SP, * : changed SP_{sprint}, ** : changed SP_{last}

Hence, we exclude only the work items that have the initial SP assigned later than one hour after they were assigned to a sprint.

3.3 Approach

In this section, we describe our approach to answer each of our research questions.

3.3.1 (RQ1) To what degree do Story Points change?

To address RQ1, we investigate the degree that the SP were changed after the work item was assigned to a sprint. We investigate this in three aspects, i.e., (1) the number of work items with SP changes in a project (i.e., the *SP change prevalence*), (2) how often the SP were changed for each work item (i.e., the *SP change frequency*), and (3) how large was the change of SP (i.e., the *SP size change*). To examine the *SP change prevalence*, we count the number of work items with changed SP (i.e., the work items that have at least one SP change event that occurred later than one hour after the work item was assigned to a sprint). To examine the *SP change frequency*, we count the number of SP change events in the activity log of a work item with changed SP. Note that we do not count the event where the SP value was initially assigned (i.e., the SP value was changed from `null` to a number). For example, in Fig. 3, the *SP change frequency* of work item #1 and #2 is zero, while the *SP change frequency* in work item #3 is one. To examine the *SP size change* of the work items with changed SP, we calculate the relative difference between changed SP_{sprint} and changed SP_{last} ($\frac{changedSP_{\text{last}} - changedSP_{\text{sprint}}}{changedSP_{\text{sprint}}} \times 100\%$) of a work item. For example, in Fig. 3, the *SP size change* of work item #3 is 200% (i.e., $\frac{3-1}{1} \times 100\%$).

In addition, to better understand the nature of SP changes in the studied projects, we examine SP changes in other three aspects, i.e., (1) how long (in days) from the time when the work item was assigned to a sprint until the SP were changed, (2) whether the person who change the SP is the work item reporter or assignee, and (3) the number of SP estimators in the project. To do so, we examine the history log of the work items to extract the time that SP were changed, the user who change the SP, and count the SP estimators.

(Porru et al. 2016) suggested that the type of work item can be associated with SP estimation. Hence, we examine whether the *SP change prevalence* is different across the different types of work items. To do so, we use Kruskal-Wallis Rank Sum tests to determine if the *SP change prevalence* is statistically different in different types of work item. The Kruskal-Wallis Rank Sum test is a non-parametric statistical test for testing whether the distribution of the multiple sample datasets are similar (Kruskal and Wallis 1952). The statistically significant result of the Kruskal-Wallis Rank Sum test indicates that the *SP change prevalence* is not different for the work items of different types.

3.3.2 (RQ2) Do changed Story Points better reflect the development time than the ones that remain unchanged?

To address RQ2, we model the relationship between SP and the development time. Then, we analyze the models to determine whether the SP can reflect the development time.

Modelling the relationship We construct Linear Mixed-Effects models to capture the relationship between SP and development time. To do so, we first extract the development time (in hours) from the activity log of a work item. The development time is the time when the work item status was set as '*in progress*' until it was set as '*done*', '*completed*', or '*fixed*'. It is possible that a work item may be estimated differently given the different contexts, i.e., priority or types of work item (Porru et al. 2016; Usman et al. 2018). In addition, as our studied projects have a long lifespan ($>$ 5 years), the practices may be changed over time (Gralha et al. 2018). For example, the SP value of 2 in 2015 may represent the development time of 2 hours, while the SP value of 2 in 2022 may represent 4 hours. Therefore, we use the priority, the types of work item, and the year that the SP were estimated as control variables in our models. When fitting the relationship, the SP value is assigned as an independent variable (i.e., fixed effects). The priority, type of work item, and the year that the SP were estimated are assigned as random effects. Finally, the development time is assigned as the response variable.

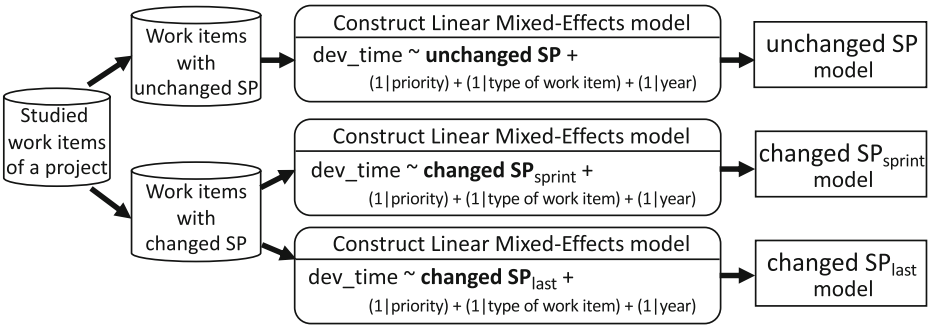


Fig. 4 An overview of our RQ2 models construction

In this RQ, we build three models for each kind of SP, i.e., (1) the unchanged SP, (2) the changed SP_{sprint}, and (3) the changed SP_{last}. Figure 4 shows an overview of our model construction approach. The unchanged SP model uses the SP of the work items with unchanged SP as an independent variable. The changed SP_{sprint} model uses the SP of the work items with changed SP at the time when the work items were assigned to a sprint as an independent variable. The changed SP_{last} model uses the last SP of the work items with changed SP as an independent variable.

Analyzing the models We use two methods to assess the relationship between SP and the development time, i.e., (1) assessing the goodness of fit of the models and (2) measuring the accuracy of the models in predicting the development time. We describe the two methods below.

To assess the goodness of fit of the models, we use a Log-Likelihood Ratio (LR) test to compare our models with the null models (i.e., $dev\_time \sim 1$). Note that we build two null models, i.e., one uses the work items with changed SP and another one uses the work items with unchanged SP. The LR test is used to estimate LR χ^2 and its significance level (p-value). A large LR χ^2 value indicates that the fit of our model is better than the null model. Then, we assess the relationship between SP and the development time using Wald χ^2 statistics. The larger the Wald χ^2 value is, the stronger the relationship between the SP and the development time is. Finally, to compare the Wald χ^2 of the SP across the three models, we normalize the Wald χ^2 value of the SP by the LR χ^2 of the model.

To evaluate the accuracy of the models, we measure the Standardized Accuracy (SA). In particular, SA measures how well our models can estimate the development time in comparison to random guessing (Sarro et al. 2016; Tawosi et al. 2021). We measure SA using a calculation of $\left(1 - \frac{MAE_{our}}{MAE_{guess}}\right) \times 100$, where MAE is the Mean Absolute Error in predicting the development time. The MAE_{our} is the MAE of our models and MAE_{guess} is the MAE of random guessing. To calculate MAE, we use a calculation of $\frac{1}{|N|} \sum_{i \in N} |estimatedDevTime_i - actualDevTime_i|$, where N is a set of studied work items. The higher SA value indicates that the model can estimate the development time closer to the actual development time.

3.3.3 (RQ3) To what extent are Story Points changes are associated with information changes?

To address RQ3, we perform both quantitative and qualitative analyses. For the quantitative analysis, we examine the association between the number of work items with changed SP and the number of work items with changed information. For the qualitative analysis, we manually identify what types of changes were made to the summary and description of work items. Then, to investigate the types of information changes that might lead to SP changes, we examine the types of information changes that are found in the work items with changed SP and the work items with unchanged SP. Below, we describe our approach for identifying the work items with information changes, the quantitative analysis (Section 3.3.3.1), and the qualitative analysis (Section 3.3.3.2).

To identify the work items with information changes, we examine the activity log of the work items. We identify that a work item has information changes when its activity log has at least one event of changing the summary or description after the work item was assigned

to a sprint. As the first hour after sprint planning could be the documentation period (see Section 3.2, DP-3), we do not consider the change events that occur within one hour after the work item was assigned to a sprint since those changes may correspond to the recent sprint planning activity.

Quantitative Analysis We examine the association between the number of work items with changed SP and the number of work items with changed information. Firstly, we construct a contingency table for the number of work items with changed SP and the number of work items with changed information. Then, we use the one-sided Fisher's Exact test to test whether the work items with changed SP have information changes more often than the work items with unchanged SP. The Fisher's Exact Test is a statistical analysis to determine an association between two categorical variables (Fisher 1992). In our case, the Fisher's Exact test determines the odds ratios and their statistical significance (p-value). In particular, an odds ratio greater than one indicates that the work items with changed SP have information changes more often than the work items with unchanged SP.

Qualitative Analysis To better understand what types of information changes made to the work items, we manually examine the changes occurred in the summary and description of work items between the version at the time when the work items were assigned to a sprint (information$_{@\text{sprint}}$) and the version at the time when the work items are closed (information$_{@\text{close}}$). Then, we identify the types of information changes using an open coding approach (Charmaz 2014). Below, we describe three main steps for our qualitative analysis, i.e., (1) open coding, (2) validation, and (3) categorization. Figure 5 shows an overview of our qualitative analysis.

Open coding. Similar to prior studies (Madampe et al. 2020; El Zanaty et al. 2018; Rigby and Storey 2011), we apply an open coding approach to discover the codes, i.e., the types of information change. To build the list of codes, we use DM which is our largest studied dataset. In the coding session, the first and second authors of this paper code the types of information changes for the randomly selected work items. The focused question for coding is "what types of information changes were made to the summary and description?". For each work item, the two coders examine the text differences between the information$_{@\text{sprint}}$ and the information$_{@\text{close}}$ in a co-located session. Then, the coders discuss until both agree on the code. In some cases, contextual information is needed to better understand the reasons for the changes. Hence, the comment history is sometimes considered in order to understand the context. The manual coding is conducted for a batch of 50 randomly selected work items until we reach the *saturation* stage, i.e., no new code is discovered for the entire set of 50 work items. We reach the *saturation* stage after the manual coding for 250 work items. Then, we revisit the whole 250 work items for another two passes since the late-discovered

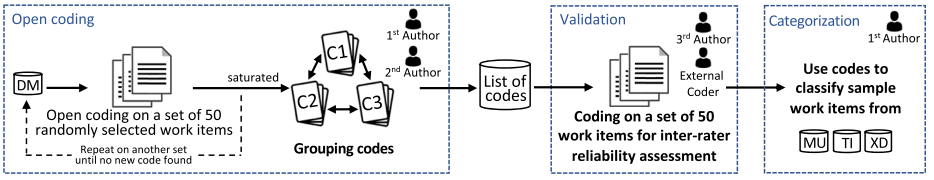


Fig. 5 The approach of our qualitative analysis to discover and validate the codes

Table 2 The codes of the information change

Code	Description
(C1) Changing scope	
Task adding	Tasks or subtasks of a work item were added. The task can be added in the form of bullet points or described in a paragraph.
Task changing	Tasks or subtasks of a work item were changed from one to another.
Task removing	Tasks or subtasks of a work item were removed.
(C2) Clarification	
Details improving	The detail of the existing tasks was modified to improve the clarity of the tasks.
Rephrasing	The detail of the work items was changed to improve readability.
Text formatting	The detail of the work items was reformatted, including paragraph adjustment, spacing, or adding and editing JIRA text formatting notation.
(C3) Others	
Work logging	A note for completed work.
Typo fixing	Typo errors were fixed or a missing conjunction word was added.
Work item organizing	A prefix or suffix in the work item summary was added or removed to indicate a group of work items.

codes may be applied to the earlier work items. Finally, we make a cohesive group for codes based on their common properties. Table 2 provides an overview of our codes in this study.

Validation. We validate our codes that are discovered in the open coding session by assessing the agreements on our coding results with two additional coders, i.e., the third author and an external coder. The external coder is a senior software engineer who has been working in the software industry for ten years. To validate the code, we randomly select 50 work items from 250 coded work items. We ask the additional coders to independently categorize those 50 work items based on our codes (see Table 2). Then, we measure the inter-rater reliability between all four coders using percent agreement and Cohen's kappa coefficient (Syed and Nelson 2015; McHugh 2012). The higher percent agreement and Cohen's kappa coefficient indicate the higher agreement among all four coders on the codes. Based on the randomly selected 50 work items, the agreement of the four coders reached the average percent agreement of 86% and the average Cohen's Kappa value of 0.83, indicating that our coding results achieve strong agreements across the four coders (McHugh 2012; Syed and Nelson 2015).

Categorization. Once we obtain the codes derived from the DM dataset, we apply our codes on three other large datasets, i.e., MU, TI, and XD. Since our studied datasets are too large to categorize entirely, we randomly select a statistically representative sample with a 95% confidence level and ten confidence interval. We select a representative sample from the work items with changed SP and another representative sample from the work items with unchanged SP. Finally, we manually categorize the sampled work items using our codes in Table 2. Note that we do not find new code that emerged while categorizing the work items in MU, TI, and XD.

3.3.4 (RQ4) Can we predict whether a work item will have Story Points changes?

To address RQ4, we build classifiers based on 41 metrics that capture various characteristics of work items. We also investigate the most influential metrics in our classifiers to better

understand what characteristics of work items can be used to predict future SP changes. Below, we describe our approaches of constructing classifiers (Section 3.3.4.1), evaluating classifiers (Section 3.3.4.2), and examining the influential metrics (Section 3.3.4.3).

Constructing Classifiers To construct classifiers, we first use 41 metrics to measure the characteristics of a work item. Then, we mitigate the colinearity of the metrics using the correlation analysis. After that, we construct our classifiers based on the remaining metrics. We describe the details of our classifiers construction approach (CC-1 to CC-4) below.

(CC-1) Measuring Characteristics. We measure the characteristics of work items using 41 metrics which are grouped into six dimensions, i.e., (1) activity, (2) completeness, (3) experience, (4) collaboration, (5) readability, and (6) text. Table 3 listed all 41 metrics we used in this study.

Activity dimension captures the activities that occurred in a work item before or at the time when the work item was assigned to a sprint. Our intuition is that the work items with many information changes are more likely to have SP changes. Our RQ3 also shows that SP were changed along with information changes. Hence, we measure how often the information described in the summary and description of a work item was changed (*activity-infochange-count*); and how large is the change by counting the number of words that were added or deleted (*activity-word-count*). We count the number of comments that were posted before the work item was assigned to a sprint (*activity-comment-count*). We also consider the SP value at the time when the work item was assigned to a sprint (*activity-sprint-sp*), the initial SP value (*activity-initial-sp*), and the size of the change from *activity-initial-sp* to *activity-sprint-sp* (*activity-sp-size-change*).

Collaboration dimension measures the activities that the reporter of the work item under the study has involved in the community. Our intuition is that the SP estimated by the reporter that has a closer connection with other developers are less likely to be changed. (Hoda and Murugesan 2016) also suggest that effective communication leads to a better effort estimation. To extract the collaboration metrics, similar to prior studies (Zanetti et al. 2013; Fan et al. 2018), we examine a collaboration network. To do so, we first construct the collaboration network where nodes are the developers in prior work items and the directed edges are the comments. For example, $A \rightarrow B$ indicates that developer A posted a comment in a work item created by developer B. Similar to (Zanetti et al. 2013), we consider only the recent comments that occurred within the past 30 days before the creation date of the work item under study. Note that we construct a collaboration network for the reporter of each work item. Similar to prior work (Zanetti et al. 2013), we measure eight network metrics for a reporter, i.e., *collab-in-degree*, *collab-out-degree*, *collab-total-degree*, *collab-kcoreness*, *collab-clustering-coefficient*, *collab-closeness-centrality*, *collab-betweenness-centrality*, *collab-eigenvector-centrality*. The higher the value of the network metric is, the closer the connection to the community that the reporter has.

Completeness dimension measures the completeness of technical information provided in a work item. In RQ3, we observe that information in a work item was changed to include more technical information. (Zimmermann et al. 2010) also show that technical information is important for bug fixing. Therefore, we examine whether or not the description contains stack traces (*has-stack*), code samples (*has-code*), patches (*has-patch*), test cases (*has-testcases*), steps to reproduce (*has-steps*), expected behaviors (*has-expected-behaviors*), observed behaviors (*has-observed-behaviors*), hyperlink (*has-link*), bullets of task (*has-bullets-task*), and attachment (*has-attachment*). Similar to prior studies (Fan et al. 2018; Chaparro et al. 2017), we use regular expressions to extract the technical information.

Table 3 An overview of our metrics that we use to measure the work item characteristics

	Metric	Description
Activity	*activity-infochange-count* *activity-word-count* *activity-comment-count* *activity-sprint-sp* *activity-initial-sp* *activity-sp-size-change*	Activity dimension captures the activities occur in a work item before or at the time when it was assigned to a sprint.
Collaboration	*collab-in-degree* *collab-out-degree* *collab-total-degree* *collab-kcoreness* *collab-clustering-coefficient* *collab-closeness-centrality* *collab-betweenness-centrality* *collab-eigenvector-centrality*	Collaboration dimension measures the activities that a work item reporter has involved in the community.
Completeness	*has-stack*, *has-code*, *has-patch*, *has-testcases*, *has-steps*, *has-expected-behaviors*, *has-observed-behaviors*, *has-link*, *has-bullets-task*, *has-attachment*	Completeness dimension measures the completeness of technical information provided in a work item.
Experience	*reporter-workitem-num* *reporter-recent-workitem-num* *reporter-stable-sp-rate* *assignee-workitem-num* *assignee-recent-workitem-num* *assignee-stable-sp-rate*	Experience dimension measures the number of work items in which the reporter and assignee of the work item under study were involved.
Readability	*read-flesch*, *read-fog*, *read-lix*, *read-kincaid*, *read-ari*, *read-coleman-liau*, *read-smog*	Readability dimension measures the text complexity in the summary and description of work items.
Text	*text-multinomial* *text-bernoulli* *text-gaussian* *text-complement*	Text dimension estimates the textual content score, i.e., likelihood that a work item will have SP changes based on textual content

For *has-attachment*, we use the JIRA REST API to check whether a work item has attached files or not.

Experience dimension measures the number of work items that the reporter and assignee of the work item under study were involved. Prior studies show that developer experience is important in effort estimation (Usman and Britto 2016; Haugen 2006; Masood et al. 2022).

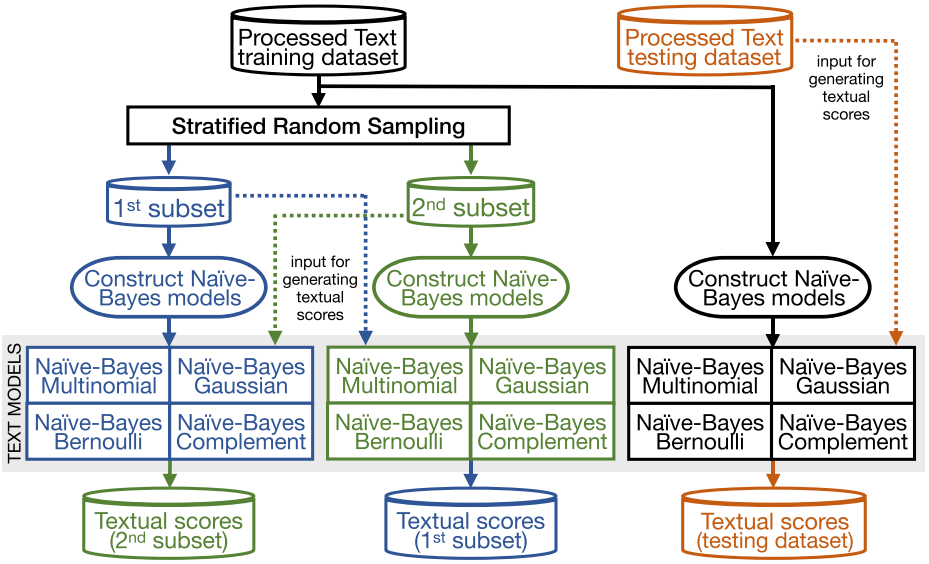


Fig. 6 An overview of the calculation approach for the text dimension metrics

Similar to prior work (Fan et al. 2018), we measure the experience of work item reporters by counting the number of prior reported work items (*reporter-workitem-num*), the number of prior reported work items within the past 90 days (*reporter-recent-workitem-num*), and the rate of prior reported work items with unchanged SP (*reporter-stable-sp-rate*). We also measure the experience of the work item assignee by counting the number of prior assigned work items (*assignee-workitem-num*), the number of prior assigned work items within the past 90 days (*assignee-recent-workitem-num*), and the rate of prior assigned work items with unchanged SP (*assignee-stable-sp-rate*).

Readability dimension measures the text complexity in the summary and description of work items. Prior work has shown that readability can indicate the quality and the validity of the work items (Zimmermann et al. 2010; Fan et al. 2018). Similar to prior work (Fan et al. 2018), we use seven readability metrics, i.e., Flesch (*read-flesch*, (Flesch 1948)), Fog (*read-fog*, (Gunning 1952)), Lix (*read-lix*, (Jonathan Anderson 1983)), Flesch-Kincaid (*read-kincaid*, (Kincaid et al. 1975)), Automated Readability Index (*read-ari*, (Senter and E.A.Smith 1967)), Coleman-Liau (*read-coleman-liau*, (Coleman and Liau 1975)), and SMOG (*read-smog*, (Mc Laughlin 1969)). These metrics indicate the education level required to comprehend the text based on the number of syllables, words, and length of sentences. The higher the readability score is, the more difficult to understand the text. Note that we concatenate the summary and description into one text before calculating the readability metrics.

Text dimension estimates the textual content score, i.e., the likelihood that a work item will have SP changes based on textual content. Recent studies have shown that the content of a work item can be used to estimate the SP (Choetkiertikul et al. 2019). To measure text metrics, we first concatenate the summary and description of work items into one text. Then, we tokenize, remove stop words, and stem the remaining words using Python Natural

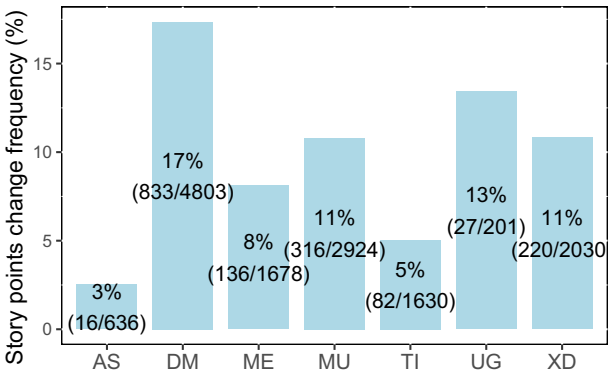


Fig. 7 The number of work items where the SP were changed for each studied project

Language Toolkit (NLTK).[11] Finally, we build Naive-Bayes models to estimate the textual content score of a work item. Figure 6 shows an overview of our approach to calculate text metrics for training and testing datasets. For the training dataset, we use the stratified random sampling technique to separate the dataset into two subsets while maintaining the proportion of work items with changed SP. Then, we use the first subset of the training dataset to build Naive-Bayes models to generate textual scores for the second subset of the training dataset. Similarly, we use the second subset to build Naive-Bayes models for the first subset. For the testing dataset, we use the whole training dataset to build Naive-Bayes models to generate the textual scores. In this work, we use four Naive-Bayes algorithms, i.e., Multinomial Naive-Bayes (*text-multinomial*, (McCallum and Nigam 1998)), Bernoulli Naive-Bayes (*text-bernoulli*, (McCallum and Nigam 1998)), Gaussian Naive-Bayes (*text-gaussian*, (Geiger and Heckerman 1994)), and Complement Naive-Bayes (*text-complement*, (Rennie et al. 2003)).

(CC-2) Analyzing Metrics Correlation. Prior study found that highly-correlated metrics may interfere with the estimated relationship between the explanatory variables and response variable in our classifiers (Tantithamthavorn et al. 2018). Therefore, we use the `AutoSpearman` function of R Rnalytica package (Jiarpakdee et al. 2018) to perform a correlation analysis and automatically remove the highly-correlated metrics. Specifically, `AutoSpearman` first measures the Spearman rank correlation (ρ) for each pair of metrics. Then, for each pair of highly-correlated metrics ($|\rho| > 0.7$), it selects one metric that shares the least correlation with other metrics that are not in the pair. Then, the other metric in the pair is removed.

(CC-3) Mitigating Imbalance Classes. Since our datasets are imbalanced (See Fig. 7), we applied a class re-balancing technique (i.e., SMOTE (Chawla et al. 2002)) to re-sample the minority class, i.e., the work items with SP changes. To do so, we use `SMOTE` function of the DMwR R package (Torgo 2010). In addition, we use Differential Evolution (DE) to optimize the number of nearest neighbors (K) and the percentage of the minority class to be resampled (Mullen et al. 2011). Note that we apply the class re-balancing only on the training dataset.

[11] https://www.nltk.org/

Table 4 The metrics of the classifier for each studied project

Dimensions	Characteristic	AS	DM	ME	MU	TI	UG	XD
Activity	*activity-infochange-count*	•	•	•	•	•	•	•
	activity-word-count		•	•	•	•	•	
	activity-comment-count						•	
	activity-sprint-sp	•			•	•		•
	activity-initial-sp		•	•			•	
	activity-sp-size-change		•	•	•	•	•	•
Collaboration	*collab-in-degree*			•	•	•	•	
	collab-out-degree		•					•
	collab-total-degree							
	collab-kcoreness		•	•	•			•
	collab-clustering-coefficient							
	collab-closeness-centrality		•	•	•	•	•	•
	collab-betweenness-centrality		•	•	•	•	•	
	collab-eigenvector-centrality					•		
Completeness	*has-stack*	•	•	•	•	•	•	•
	has-code	•	•	•	•	•	•	•
	has-patch							
	has-testcases	•	•	•	•	•		•
	has-steps	•	•	•	•	•		•
	has-expected-behaviors	•				•		
	has-observed-behaviors					•		
	has-link	•	•	•	•	•	•	•
	has-bullets-task	•	•	•	•	•	•	•
	has-attachment	•	•	•	•	•		•
Experience	*reporter-workitem-num*		•	•	•	•	•	•
	reporter-recent-workitem-num	•	•	•	•		•	•
	reporter-stable-sp-rate	•	•	•	•	•	•	•
	assignee-workitem-num	•	•	•				
	assignee-recent-workitem-num	•	•	•	•	•	•	•
	assignee-stable-sp-rate		•	•	•	•	•	•
Readability	*read-flesch*		•					
	read-fog							
	read-lix	•	•		•		•	•
	read-kincaid							
	read-ari							
	read-coleman-liau	•		•	•	•		•
	read-smog			•		•		
Text	*text-multinomial*				•	•		•
	text-bernoulli	•	•	•	•	•	•	•

Table 4 (continued)

Dimensions	Characteristic	AS	DM	ME	MU	TI	UG	XD
	text-gaussian		•	•	•	•	•	•
	text-complement	•	•	•			•	

(CC-4) Constructing Classifiers. We use the remaining metrics surviving from our correlation analysis to construct our classifiers. Table 4 shows the list of surviving metrics of each studied project. To construct our classifiers, we adopt four classification techniques, i.e., non-linear Logistic Regression (LR), Random Forest (RF), Support Vector Machine (SVM), and Classification And Regression Tree (CART) that were used in prior effort estimation studies (Xia et al. 2020; Kocaguneli et al. 2011; Corazza et al. 2013; Cerpa et al. 2010). We assign our metrics as explanatory variables. The response variable is assigned as `TRUE` for the work item that will have SP changes later than one hour after it was assigned to a sprint, and `FALSE` otherwise. We use the `lrm` function of the rms R package (Harrell Jr. 2019) to construct the LR classifiers, `randomForest` function of the randomForest R package (Liaw and Wiener 2018) to construct the RF classifiers, `svm` function of the e1071 R package (Meyer et al. 2019) to construct the SVM classifiers, and `rpart` function of the rpart R package to construct the CART classifiers (Therneau et al. 2015).

To improve the performance of our classifiers, we fine-tune our classifiers using Differential Evolution (DE). First, we generate a fine-tuning dataset from the training dataset. To do so, we use stratified random sampling to generate the fine-tuning dataset with the size of 20% of the training dataset while preserving the proportion of the work items with changed SP. Then, we use DE to optimize the number of trees in RF classifiers, the cost of constraint violation (C) and curvature weight of the decision boundary (γ) in SVM, and the complexity of the decision tree (CP) in CART. To do so, we use the `DEoptim` function of the DEoptim R package (Mullen et al. 2011).

For LR classifiers, as similar to prior work (Thongtanunam et al. 2017; McIntosh and Kamei 2017; Ruangwan et al. 2018), we allocate additional degrees of freedom to the explanatory variables based on the Spearman multiple ρ^2, i.e., five and three degrees of freedom to explanatory variables that have strong ($\rho^2 > 0.3$) and moderate ($0.15 < \rho^2 \leq 0.3$) potential for a non-linear relationship, respectively.

Evaluating Classifiers To evaluate our classifier, we compare our classifiers against four baselines based on five performance measures. We describe our evaluation approach (EC-1 to EC-4) below.

(EC-1) Measuring Classifiers Performance. We evaluate the performance of a classifier using the Area Under the receiver operator characteristic Curve (AUC), Balanced Accuracy, Recall, False Alarm Rate, and Distance to Heaven. **AUC** measures an area below the curve of the true positive rate $\left(\frac{TP}{TP+FN}\right)$ against the false positive rate $\left(\frac{FP}{FP+TN}\right)$ (Hanley and McNeil 1982). An AUC value greater than 0.5 indicates that our classifier performs better than random guessing. An AUC value of 1 indicates that our classifier well discriminate between the work items that will have SP changes and those that will not have SP changes. **Balanced Accuracy (BACC)** measures the accuracy for an imbalanced dataset (Velez et al. 2007) by normalizing the true positive and true negative rates

$\left(\frac{1}{2}\left(\frac{TP}{TP+FN}+\frac{TN}{TN+FP}\right)\right)$. **Recall** measures how many work items that will have SP changes that can be identified $\left(\frac{TP}{TP+FN}\right)$. **False Alarm Rate (FAR)** measures the proportion between the number of work items with unchanged SP that are mis-classified as having changed SP $\left(\frac{FP}{FP+TN}\right)$. A lower FAR value indicates that the fewer work items with unchanged SP are mis-classified as having changed SP. **Distance to Heaven (D2H)** is a combination of recall and FAR, suggested by (Agrawal et al. 2019). We measure D2H using a calculation of $\sqrt{\frac{(1-Recall)^2+(0-FAR)^2}{2}}$ (Agrawal et al. 2019). A lower D2H value indicates that our approach achieves a better classification, i.e., high Recall and low FAR value. We used Recall, FAR, and D2H because these measures are recommended to use when the data is imbalanced (Agrawal et al. 2019). We do not use precision and F1-score because they may not reflect the performance of classifiers in the imbalanced data (Menzies et al. 2007).

Prior studies have reported that imbalanced datasets can affect the threshold-depending performance measures, e.g., Precision, Recall, F1-score (Tantithamthavorn and Hassan 2018; Tantithamthavorn et al. 2018). Hence, we determine the optimal threshold based on the training dataset. To do so, we use our classifier to estimate the probability that a work item in the training dataset will have SP changes. We define that the optimal threshold is the average value between the first quartile of the probabilities of the work items that will have SP changes and the third quartile of the probabilities of the work items that will *not* have SP changes (i.e., $\frac{1}{2}(Q_{1\_changedSP}+Q_{3\_unchangedSP})$). We determine the optimal threshold for each classifier and each training dataset.

(EC-2) Baseline Approaches. We compare the performance of our classifiers against four baselines, which are described below:

One-Rule Classification algorithm (OneR): OneR is a simple classification algorithm for supervised learning. OneR selects one best metrics (based on the training dataset) to predict the outcome with the smallest error (Holte 1993; von Jouanne-Diedrich 2017). Although, OneR is not known for its good classification performance, we used it as our lower-bound performance baseline. This will also evaluate whether our classifier built based on 41 metrics outperforms a simple classifier built using only one best metric. Note that we do not compare our classifiers with OneR in terms of AUC since OneR does not produce likelihood estimates.

Majority Vote: Majority Vote is a classification technique which predicts the target class based on the majority vote of the metrics. We used Majority Vote as our baseline to evaluate whether our classifier can outperform the classifier that considers similar set of metrics but using a different algorithm. In our Majority Vote approach, each metric votes whether a work item will have SP changes based on a threshold of the metric value. For each metric, the threshold is the metric value that achieves the highest recall rate based on the training dataset. Then, the approach predicts the outcome based on the majority votes of all metrics.

Latent Dirichlet Allocation (LDA)-based classifier: A recent study showed that LDA-based classifiers can be used to predict the SP (Tawosi et al. 2022). Hence, we used LDA-based classifier as our baseline to evaluate whether our classifier can outperform a classifier built based on the topics extracted from natural language text. LDA is an unsupervised technique for generating topics from multiple text documents based on independent word distributions (Blei et al. 2003). We first generate k topics based on the summary and description of work items. We use Differential Evolution (DE) to determine the optimal number of topics (k) based on the training dataset (Mullen et al. 2011). Then, we built a classifier based on the generated LDA topics as features of work items. Specifically, for each work item, we use the topic scores generated from LDA as the input for the classifier. We

experiment this approach with three classification techniques, i.e., Random Forest, Classification And Regression Tree, and Naive-Bayes. We find that using Random Forest provides the best performance among the three techniques. Therefore, we use LDA with Random Forest (LDA-RF) as a baseline.

Text-based Classifier: Prior studies show that the summary and description of the work items could be used to predict the SP (Choetkiertikul et al. 2019; Porru et al. 2016). Hence, we use a text-based classifier as our baseline to evaluate whether our classifier can outperform a classifier built using the summary and description of work items, which known to be an effective predictor of SP. We construct a classifier to predict the future SP changes based on the summary and description of work items. We experiment this approach with four Naive-Bayes techniques, i.e., Multinomial, Gaussian, Bernoulli, and Complement. We find that using Naive-Bayes Bernoulli provides the best performance among the four techniques. Therefore, we use Text classifier with Naive-Bayes Bernoulli as a baseline.

(EC-3) Validating Performance. We use the out-of-sample bootstrap validation technique to validate our classifier while mitigating bias in performance estimation (Efron 1983; Harrell Jr 2015; Tantithamthavorn et al. 2017). To do so, we first generate a bootstrap sample, i.e., a dataset that is sampled with replacement from the original dataset. The bootstrap sample has the same size as the original dataset. Then, we use this bootstrap sample as a *training* dataset to construct the classifier. The instances in the original dataset that do not appear in the bootstrap sample (i.e., the training dataset) will be used as a *testing* dataset to evaluate the classifier. Finally, we repeat this process 100 times.

(EC-4) Comparing Performance. To confirm whether our classifier is better than the baselines, we perform a statistical analysis as follow. First, for each of the 100 bootstrap samples, we measure the performance of our classifier and the baselines. Hence, each approach will have five performance distributions (i.e., AUC, BACC, Recall, FAR, and D2H) estimated based on the 100 bootstrap samples. Then, for each performance measure, we use a one-sided Wilcoxon signed-rank test (Wilcoxon 1992) to compare the performance distribution of our classifier against the performance distribution of the baselines. We also measure the effect size (r), i.e., the magnitude of the difference between two distributions using a calculation of $r = \frac{Z}{\sqrt{n}}$, where Z is a statistic Z-score and n is the number of work items in the testing dataset (Tomczak and Tomczak 2014). Based on the rule of thumb, $r \geq 0.8$ is considered as large, $0.5 \leq r < 0.8$ is medium, and $0.2 \leq r < 0.5$ is small, otherwise negligible (Sawilowsky 2009). A larger and significant effect size ($p < 0.05$) represents a better performance of our classifier over the baselines. We used Z-score statistic because we used a paired difference test (i.e., the Wilcoxon signed-rank test). We did not use the commonly-used Cohen's D (Cohen 2013) nor Cliff's δ (Macbeth et al. 2011) to measure the effect size because both methods are not based on the assumption that the samples are matched.

Examining the Influence of Metrics To examine the influence of metrics, for each project, we analyze each of the 100 classifiers that are constructed using the bootstrap samples. Since we found that LR classifiers achieve the best performance, we examine the influential metrics in our LR classifiers. We use Wald χ^2 statistics to determine the explanatory power (χ^2) of each metric that contributes to the classifier and the overall χ^2 of the classifier. Then, we normalize the χ^2 value of the metrics by the overall χ^2 of the classifier. Finally, we rank the metrics based on the normalized χ^2 values across all 100 classifiers. We use the non-parametric Scott-Knott Effect Size Difference (SK-ESD) test to cluster the metrics into statistically distinct ranks (Tantithamthavorn et al. 2017; 2019).

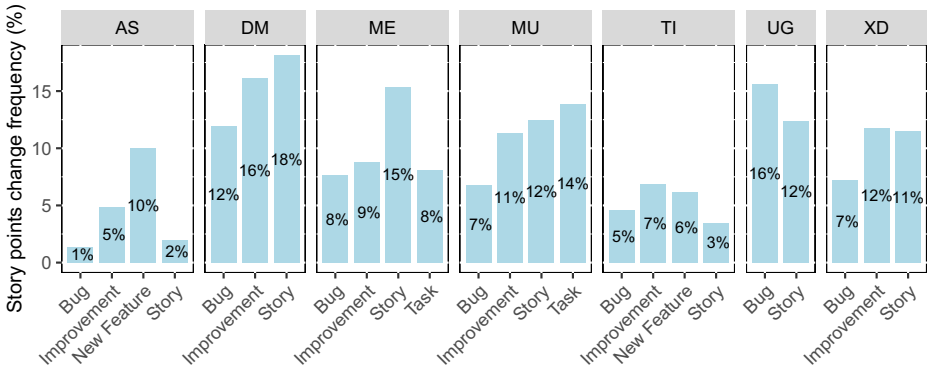


Fig. 8 The number of work items where the SP were changed for each type of work item

4 Case Study Results

Based on the email responses, the core developers of the seven studied projects inform us that (1) their projects generally allow SP to be changed, (2) they estimated the SP during sprint planning, (3) they assigned the work items to the sprint during sprint planning, and (4) their typical sprint length is 2-4 weeks. They also informed us that the work items were self-assigned and estimated by the team member(s). It is not uncommon for them to change the SP after the initial SP estimation. However, most of them inform us that they try to freeze the sprint (e.g., SP, work items allocation) once the sprint is started. If they found an unexpected complexity during the sprint execution, they may re-estimate (change) the SP of the work item. Below, we present the results of our case study with respect to our four research questions.

4.1 (RQ1) To what degree do Story Points change?

On average, 10% of the studied work items have SP changes after they were assigned to a sprint. Figure 7 presents the number of work items where the SP were changed for each studied project (i.e., *SP change prevalence*). DM has the largest number of work items with changed SP (i.e., 17%), whereas AS has the smallest number of work items with changed SP (i.e., 3%). The other five projects have work items with changed SP ranging from 5% to 13%. On average across the seven projects, we find that 10% of the studied work items have SP changes after they were assigned to a sprint, suggesting that SP were not frequently changed in our studied projects.

Figure 8 shows the number of work items with changed SP across the five types of work items. Across all studied projects, we do not find the dominant type that often has SP changes. Nevertheless, we observe that *bug* work items have changed SP less often than the other types in five projects. This might be because the information for fixing bugs are clear and more stable than the information of other types, since bugs have been validated during the bug triage process.

On average, 88% of the work items with changed SP have only one SP change event after they were assigned to a sprint. Figure 9 shows the *SP change frequency* within a work item (i.e., the number of SP change events). Note that we consider only the work items with changed SP. We find that 78%(MU)-94%(AS) of the work items have only one

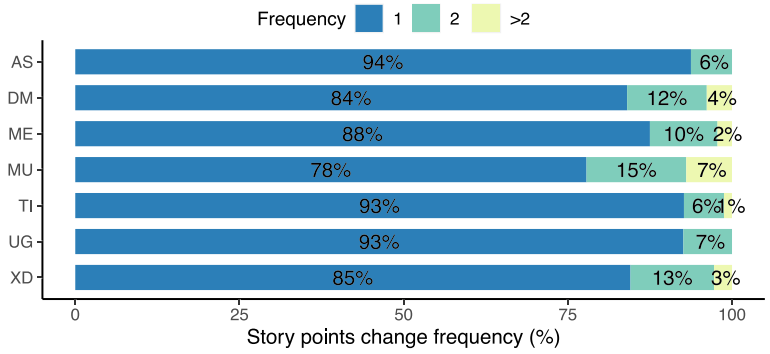


Fig. 9 The change frequency of the SP changes

SP change event. Figure 9 also shows that 6%-15% of the work items have two SP change events, and 0%-7% of the work items have more than two SP change events. This result indicates that the SP of a work item in the studied projects were not frequently changed after the work item was assigned to a sprint.

On average, 57% of work items with changed SP have their SP value increased by at least 58% relative to the SP value when they were assigned to a sprint. Table 5 shows that 8%(UG)-82%(TI) of the work items with changed SP have increased SP. At the median values, the changing size of increased SP is ranging from 58% to 100% relative to the SP value when the work item was assigned to a sprint. For example, XD-1503 aimed to refactor the duplicate code.[12] However, the SP of this work item was changed from 1 to 10 (i.e., the changing size of 1,000%). The developer also noted that the refactoring task is more complicated than expected. Table 5 also shows that 18%(TI)-92%(UG) of the work items with changed SP have decreased SP. At the median values, the changing size of decreased SP is ranging from 40% to 67% relative to the SP value when the work item was assigned to a sprint. For example, the SP of DM-9282 were changed from 10 to 6 (i.e., the changing size of 40%) since one of the subtasks was removed.[12] These results suggest that the SP tend to be increased with a relatively large size change compared to their value when the work item was assigned to a sprint.

We further examine whether the *SP size change* is associated with the type of work item using the Kruskal-Wallis Rank Sum tests. We find that the results of Kruskal-Wallis Rank Sum tests are not statistically significant in all studied projects except XD ($p \geq 0.05$). This finding indicates that the *SP size changes* are not different for the work items of different types.

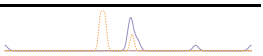
We find that the work items with changed SP typically have SP changes 1.8(AS)-16.9(DM) days after they were assigned to a sprint. We also find that these work items took 8(UG)-100.8(TI) days since they were assigned to a sprint to arrive at the close status.[13] These findings suggest that a work item typically has SP change few days after being assigned to a sprint.

In five out of the seven studied projects (except UG and XD), the majority of SP changes were made by the work items assignee, i.e., 52%(MU)-70%(ME). In particular, we find that the work items assignees changed the SP of 31%(UG)-70%(ME) of the work items with

[12] The url of Jira work items are provided in Appendix (see Section A7)

[13] A distribution plot is provided in the Appendix (see Section A3)

Table 5 Descriptive statistics of the *SP size change*. Distributions are in a log scale

Proj.	Increased				Decreased				Distribution
	#Inc.	1st Qu.	Med	3rd Qu.	#Dec.	1st Qu.	Med	3rd Qu.	Increased-Decreased (dotted line)
AS	69%	60%	60%	67%	31%	38%	40%	40%	
DM	67%	50%	100%	158%	33%	38%	50%	67%	
ME	77%	60%	67%	167%	23%	33%	40%	50%	
MU	41%	63%	100%	200%	59%	40%	50%	67%	
TI	82%	60%	63%	150%	18%	39%	60%	63%	
UG	8%	54%	58%	62%	92%	64%	67%	67%	
XD	53%	60%	100%	200%	47%	40%	50%	67%	

changed SP, while the work items reporters changed the SP of 28%(TI)-58%(DM) of the work items with changed SP. The other users who are neither reporter or assignee of the work items change the SP in 28%(ME)-41%(UG) of the work items with changed SP. These findings suggest that the SP tend to be changed by the assignees of the work items.

Findings: On average, 10% of the work items in the studied projects have SP changes after they were assigned to a sprint. 88% of them have SP changed only once. An average of 57% of them have their SP increased by at least 58% relative to the SP values when they were assigned to the sprint.
Implication: Even though SP are not frequently changed after the sprint planning, SP tend to be increased by a relatively large size change.

4.2 (RQ2) Do changed Story Points better reflect the development time than the ones that remain unchanged?

Table 6 showsthe statistics of our models that fit the relationship between the development time and (1) unchanged SP, (2) changed SP_{sprint}, and (3) changed SP_{last}. Table 6 shows that all of our models have a significant LR χ^2 ($p < 0.05$), indicating that our models fit the relationships better than the null model. We now discuss the relationship the development time with the unchanged SP and changed SP.

The unchanged SP share a stronger relationship with the development time than the changed SP. Table 6 shows that the unchanged SP have significant Wald χ^2 values in six studied projects ($p < 0.001$). On the other hand, the changed SP_{sprint} and the changed SP_{last} have significant Wald χ^2 values only in four and five projects, respectively. Table 6 also shows that the Wald χ^2 value of the unchanged SP is larger than that of the changed SP_{sprint} and the changed SP_{last} in six and five projects, respectively. Moreover, Table 7 shows that the unchanged SP models achieved a higher Standardized Accuracy (SA) than the changed SP_{sprint} and changed SP_{last} models in all seven projects. These results indicate

that the unchanged SP can better reflect the development time than the changed SP when the work item was assigned to a sprint, regardless before or after SP changes.

For the changed SP, we observe that the last SP value shares a stronger relationship with the development time than the SP value when the work item was assigned to a sprint. Table 6 and Table 7 show that the changed SP_{last} have a significantly larger Wald χ^2 value ($p < 0.01$) and achieved a higher SA than the changed SP_{sprint} in four projects. These results indicate that the last value of changed SP tends to reflect the development time better than the changed SP value when the work item was assigned to a sprint.

Finding: Based on our mixed-effects models, the unchanged SP share a stronger relationship with the development time than the changed SP. Meanwhile, the changed SP_{last} tend to share a stronger relationship with the development time than the changed SP_{sprint}.
Implication: The stable (unchanged) SP can better reflect the effort of a work item than the unstable (changed) SP.

4.3 (RQ3) To what extent are Story Points changes are associated with information changes?

Our RQ2 shows that the last value of the changed SP can better indicate the development time than the SP value at the time when the work item was assigned to a sprint. This may be in part due to the inadequate information provided in the summary and description of work items. Therefore, we perform quantitative and qualitative analyses to investigate the relationship between SP changes and information changes.

4.3.1 Quantitative Analysis

33%-93% of the work items with changed SP have information changes, while only 9%-29% of the work items with unchanged SP have information changes. Table 8 shows

Table 6 Statistics summary of our models in RQ2. The larger the χ^2 value is, the better the fit of our models and the SP values. "LRχ^2" indicates LR χ^2 of the model and "Waldχ^2" indicates the normalized Wald χ^2 of SP

	LRχ^2 of the model			Normalized Waldχ^2 of SP		
Project	USP	CSP_{sprint}	CSP_{last}	USP	CSP_{sprint}	CSP_{last}
AS	15.54 ***	8.51 **	10.69 **	0.8 ***	0.26 °	0.98 **
DM	115.16 ***	21.58 ***	41.98 ***	1.01 ***	0.71 ***	0.89 ***
ME	66.14 ***	16.21 ***	18.23 ***	0.96 ***	0.47 **	0.58 **
MU	215.79 ***	6.31 *	43.25 ***	1.05 ***	0.34 °	1.01 ***
TI	80.89 ***	20.67 ***	16.15 ***	1 ***	0.53 ***	0.44 **
UG	7.85 **	18.38 ***	13.06 ***	0 °	0.67 ***	0.21 °
XD	190.2 ***	7.2 **	7.65 **	1.05 ***	0.07 °	0.13 °

Statistical significance:*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$, ° $p \geq 0.05$
USP=unchanged SP, CSP_{sprint}=changed SP_{sprint}, CSP_{last}=changed SP_{last}

the contingency table of the work items with SP changes and the work items with changed information. We find that 33%($\frac{4}{12}$; AS)-93%($\frac{13}{14}$; UG) of the work items with changed SP have information changes, while only 9%($\frac{51}{569}$; AS)-29%($\frac{39}{135}$; UG) of the work items with unchanged SP have information changes. Table 8 also shows that the odds that a work item with changed SP will have information changes are 2.25(ME)-4.35(MU) times higher than that of a work item with unchanged SP. These results suggest that the work items with changed SP are more likely to have information changes than the work items with unchanged SP.

4.3.2 Qualitative Analysis

Table 9 shows the number of sampled work items in DM, MU, TI, and XD, and their proportion that are categorized in each type of information change. Note that only 51 work items are found with multiple categories and 16 work items were marked as "unknown" due to insufficient context knowledge. Below, we describe the three categories shown in Table 2 with our observations (Section 4.3.2.1). Finally, we discuss the difference of information changes between the work items with changed SP and the work items with unchanged SP (Section 4.3.2.2).

Types of Information Changes (C1) Changing scope. Table 9 shows that in 9%(TI)-35%(DM and MU) of the sampled work items, the summary and description were changed to update task (or subtasks).

(C1-1) *Task adding*. We observe that developers added more tasks or subtasks into the summary and description of work items after they were assigned to a sprint. The tasks can be added in the form of bullet points or described in a paragraph. The tasks that were newly added can be main tasks or subtasks. For example, in DM-12211, the description was changed and the developer explicitly noted that the scope of this work item was expanded to include additional implementation tasks, i.e., *"The scope for this ticket has been significantly expanded to include following features: 1. [feature A] 2. [feature B]"*.[12] Table 9

Table 7 [Left] The Standardized Accuracy (SA) of our models in RQ2. The higher SA value indicates a higher accuracy of our models over random guessing. [Right] The plots of SP (x-axis) and development time (y-axis)*

	Standardized Accuracy (SA)			Predicted and actual development time		
Project	USP	CSP_{sprint}	CSP_{last}	USP	CSP_{sprint}	CSP_{last}
AS	89.96	63.48	58.02			
DM	95.78	91.68	92.04			
ME	93.54	88.16	88.46			
MU	93.76	81.99	83.93			
TI	96.24	86.87	86.5			
UG	92.46	64.83	59.11			
XD	95.75	91.53	91.68			

USP=unchanged SP, CSP_{sprint}=changed SP_{sprint}, CSP_{last}=changed SP_{last}

*The blue line indicates the development time predicted by our models based on SP.

*The black dots represent the actual development time spent on the work items for the estimated SP.

Table 8 The contingency table of the work items with changed SP and the work items with changed information

	AS		DM		ME		MU		TI		UG		XD	
	C^{sp}	U^{sp}	C^{sp}	U^{sp}	C^{sp}	U^{sp}	C^{sp}	U^{sp}	C^{sp}	U^{sp}	C^{sp}	U^{sp}	C^{sp}	U^{sp}
C^{info}	4	51	311	586	41	248	101	254	26	230	13	39	72	229
U^{info}	12	569	522	3,384	95	1,294	215	2,354	56	1,318	14	135	148	1,581
Odds Ratio	3.71*		3.44***		2.25***		4.35***		2.66***		3.19**		3.36***	

C^{sp}=work items with changed SP, U^{sp}=work items with unchanged SP

C^{info}=work items with changed information, U^{info}=work items with changed information

Statistical significance: *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$, ° $p \geq 0.05$

Table 9 The number of work items for each type of information changes

Code	DM	MU	TI	XD
#Sampled (C/U)*	98 / 152	49 / 70	21 / 68	41 / 68
(C1) Changing scope	35%	35%	9%	27%
- Task adding	20%	12%	6%	5%
- Task changing	10%	18%	3%	17%
- Task removing	6%	6%	0%	6%
(C2) Clarification	51%	58%	75%	61%
- Details improving	38%	27%	55%	22%
- Rephrasing	8%	29%	16%	32%
- Text formatting	6%	3%	4%	7%
(C3) Others	24%	10%	18%	11%
- Work logging	8%	1%	0%	0%
- Typo fixing	15%	8%	9%	8%
- Work item organizing	1%	1%	9%	3%

*C=work items with changed SP

*U=work items with unchanged SP

shows that this type of information changes occurred in 5%(XD)-20%(DM) of the sampled work items.

(C1-2) *Task changing*. We observe that developers changed the existing tasks or subtasks of the work item after it was assigned to a sprint. For example, DM-814 had subtasks to clean up *"core/examples"* and *"admin/bin"* modules.[12] Then, the second subtask was changed to clean up the *"core/doc"* module instead. Table 9 shows that this type of information change occurred in 3%(TI)-18%(MU) of the sampled work items.

(C1-3) *Task removing*. We observe that developers removed subtasks from the work item after it was assigned to a sprint. For example, DM-9282 aims to implement a server-side extension that covers three subtasks.[12] Then, one of the subtasks was removed. Table 9 shows that this type of information change occurred in 0%(TI)-6%(DM, MU, and XD) of the sampled work items.

(C2) Clarification. Table 9 shows that 51%(DM)-75%(TI) of the sampled work items have an information change for clarifying the summary and description of work items.

(C2-1) *Details improving*. In this category, the summary and description of work items were modified (added, removed, or changed) to clarify the details of the existing tasks in the work item. However, the scope of work remains the same as the original one. The details also include implementation steps, steps to reproduce, observed behaviors, expected behaviors, stack traces, attachment files, or example codes. For example, TI-23156 aims to implement a new feature to automatically load the modules that are required for an API.[12] The work item description originally contained the background and description of the feature. Then, the work item description was changed in order to add test cases and the expected outcome of the feature (i.e., the modules to be loaded). Table 9 shows that this type of information change occurred in 22%(XD)-55%(TI) of the sampled work items.

(C2-2) *Rephrasing*. We observe that developers rephrased the description of tasks and other details. For this category, the changes are only for improving readability. For example, the summary of DM-2683 was rephrased from *"[module] freeze during integration test case 05"* to *"Fix case05 [module] freeze"*.[12] Table 9 shows that this type of information changes occurred in 8%(DM)-32%(XD) of the sampled work items.

(C2-3) *Text formatting*. The changes of this category are related to updating the format of existing text in the summary and description of work items. The *Text formatting* can be adding white spaces (e.g., DM-14172), line adjustment (e.g., DM-13489), or adding JIRA text formatting notation (e.g., MU-15159).[12] Table 9 shows that this type of information changes occurred in 3%(MU)-7%(XD) of the sampled work items.

(C3) Others. The changes of this category are related to logging the completed work, fixing typo errors or spelling, and organizing the work items. For *Work logging*, we observe that the developers put a note for the tasks or pull requests in the work item description that have been completed (e.g., DM-8007).[12] For *Typo fixing*, the changes can be fixing misspelling or missing conjunction (e.g., TI-2568 and DM-7267).[12] For *Work item organizing*, the changes are related to the prefix or suffix of the work item summary (for example, adding "iOS9:" into the summary of TI-19356).[12] Table 9 shows that this type of information changes occurred in 10%(MU)-24%(DM) of the sampled work items.

Information Changes and Story Points changes Now, we discuss the difference of information changes between the work items with changed SP and the work items with unchanged SP. Figure 10 shows the number of work items with changed SP and unchanged SP for each type of information change.

The work items with changed SP have information changes related to the scope of work *more* often than the work items with unchanged SP. Figure 10 shows that 14%(TI)-64%(DM) of the work items with changed SP have information changes for *Changing scope*, while only 3%(TI)-21%(MU) of the work items with unchanged SP have information changes for *Changing scope*. Specifically, figure 10 shows that 7%(TI)-41%(DM) of the work items with changed SP have information changes for *Task adding*, while only 1%(XD)-10%(MU) of the work items with unchanged SP have information changes for *Task adding*. For *Task changing* and *Task removing*, we also observe a similar trend, i.e., the work items with changed SP have the scope-related changes more often than the work items with unchanged SP.

The work items with changed SP have information changes for clarification *less* often than the work items with unchanged SP. Figure 10 shows that 29%(TI)-38%(DM and MU) of the work items with changed SP have information changes for *Clarification*, while 60%(DM)-81%(TI) of the work items with unchanged SP have information changes for *Clarification*. More specifically, Fig. 10 shows that 20%(MU and XD)-30%(DM) of the work items with changed SP have *Details improving*, while 24%(XD)-57%(TI) of the work items with unchanged SP have *Details improving*. For *Rephrasing* and *Text formatting*, we also observe a similar trend, i.e., the work items with changed SP have information changes for *Clarification* less often than the work items with unchanged SP.

The work items with changed SP tend to have information changes in the *Others* category *less* often than the work items with unchanged SP. Figure 10 shows that the work items with changed SP have information changes for *Work logging* and *Typo fixing* less often than the work items with unchanged SP in all studied projects. However, we do not observe a dominant difference of information changes for *Work item organizing* across all studied projects.

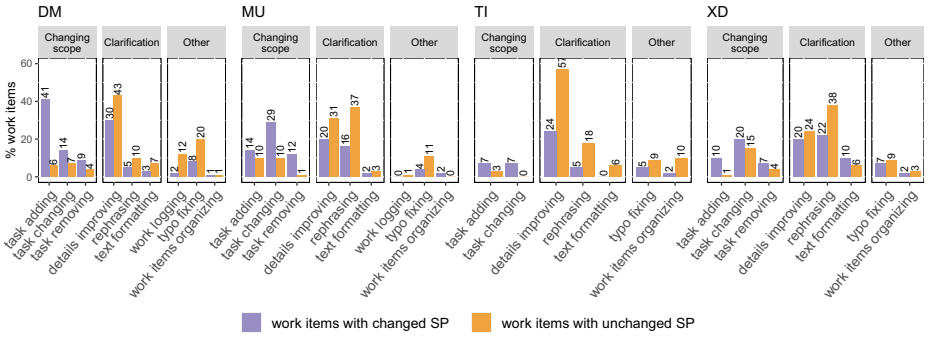


Fig. 10 The number (in percentage) of sampled work items with changed SP and with unchanged SP for each type of information change. Note that 51 work items are found with multiple types of information changes

Findings: 33%-93% of the work items with changed SP have information changes, while only 9%-29% of the work items with unchanged SP have information changes. Our manual categorization shows that 7%-41% of the work items with changed SP have information changes for updating the scope, while 60%-81% of the work items with unchanged SP have information changes for clarification.
Implication: After the sprint planning, SP changes often occur along with information changes. The work items with changed SP tend to have information changes more often than the work items with unchanged SP. Information changes for updating scope often occur in the work items that have SP changes, while information changes for clarification often occur in the work items that do not have SP changes.

4.4 (RQ4) Can we predict whether a work item will have Story Points changes?

Our RQ1 shows that *SP size change* is relatively large. Our RQ2 shows that the changed SP may not reflect the development time. Therefore, it would be worthwhile to timely predict whether the SP will be changed to helps developers be aware and better cope with the unreliability in SP estimation. Hence, we develop and measure the performance of the classifiers in predicting whether a work item will have SP changes after being assigned to a sprint (Section 4.4.1). In addition, we examine the influential metrics in the classifiers (Section 4.4.2).

4.4.1 Performance of the Classifiers

Our approach using Logistic Regression achieves an average AUC of 0.69-0.8, an average Balanced Accuracy (BACC) of 0.6-0.68, and an average Distance to Heaven (D2H) of 0.33-0.47. Figure 11 shows the performance distributions of our classifiers and baselines. We observe that the Logistic Regression (LR), Random Forest (RF), and Support Vector Machine (SVM) classifiers used in our approach achieve similar accuracy in terms of AUC across the seven studied projects (i.e., 0.69-0.8 for LR, 0.64-0.77 for RF, and 0.65-0.78 for SVM). Meanwhile, the Classification And Regression Tree (CART) classifier achieves the lowest average AUC (i.e., 0.56-0.61) among the four classification techniques. This may

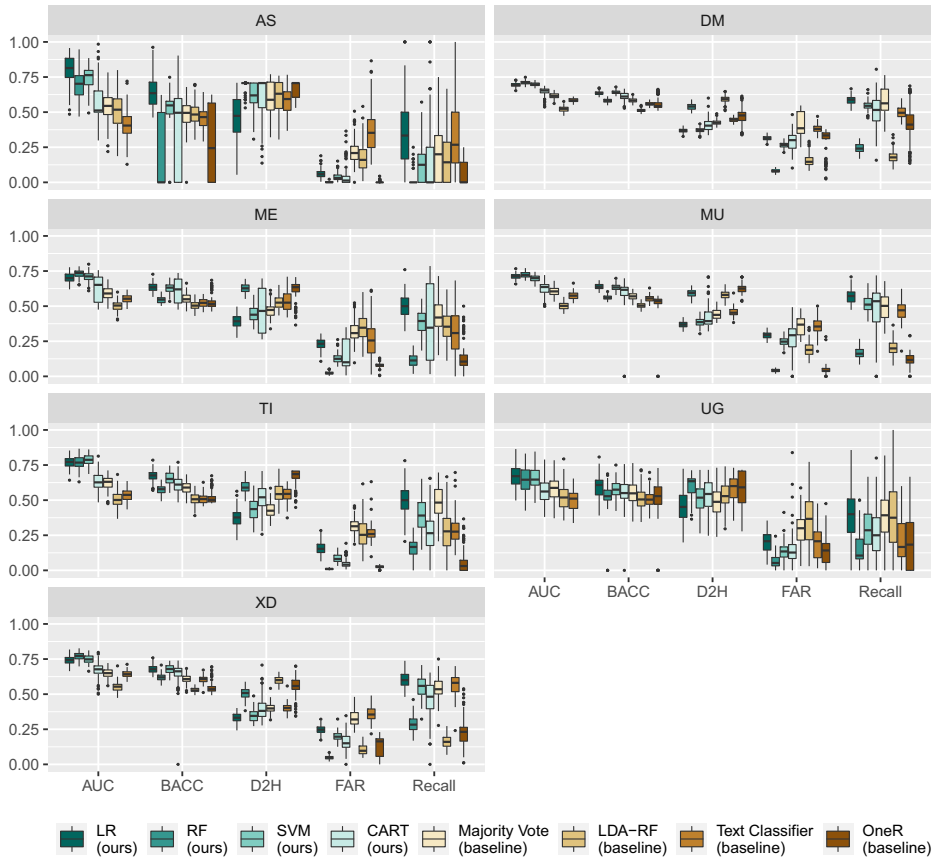


Fig. 11 The performance distributions of our classifiers (using LR, RF, SVM, and CART) and the four baselines based on 100 bootstrap samples

be because CART may produce unstable decision trees in the model. In particular, (Timofeev 2004) argued that changing several data points could lead to radical changes to CART. In our study, we built CART classifiers based on 100 bootstrap samples, which can have different data points (work items) for training. In consequence, as shown in Fig. 11, the distribution of the performance achieved by CART classifiers had a wider range of values than the other classification techniques.

In terms of other accuracy measures, we observe that LR classifier achieves the best performance, i.e., the highest average BACC and the lowest average D2H in all studied projects among the four classifiers. These results suggest that LR classifier generally performs better than other classifiers; and can accurately identify the work items that will have and will not have SP changes. Henceforth, the results of other analyses in this RQ will be reported based on LR classifiers.

Our approach achieves AUC, BACC, and D2H significantly better than the baseline approaches. Figure 11 shows that the baselines (i.e., Majority Vote, LDA, Text Classifier, and OneR) achieve an average AUC below 0.60, an average BACC below 0.56, and

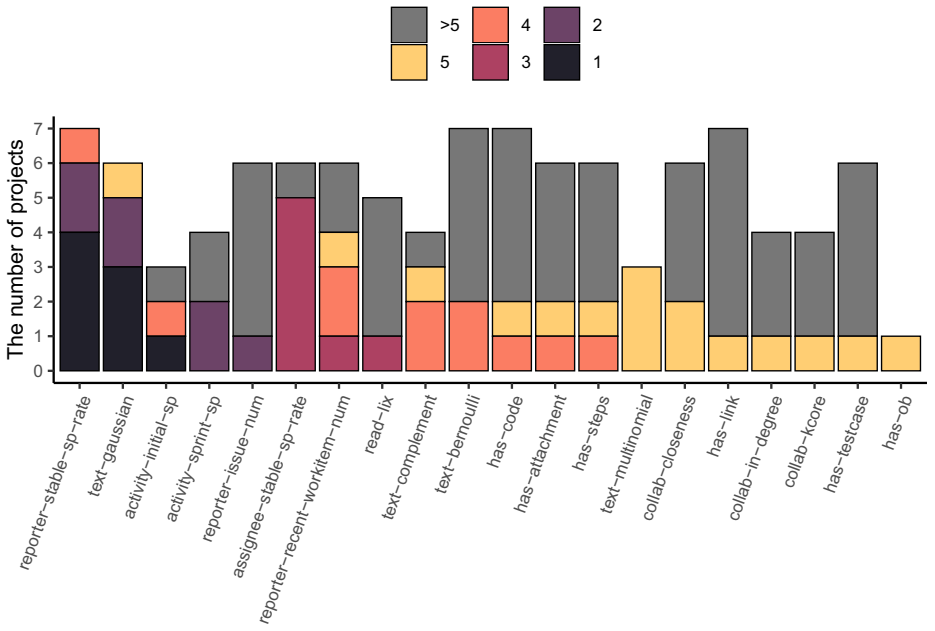


Fig. 12 The metrics that are ranked top five most influential in our Logistic Regression classifier. Noted that multiple metrics can be in the same rank based on the non-parametric Scott-Knott ESD tests

an average D2H above 0.46. The statistical tests also show that the AUC and BACC values of our approach using LR classifier are significantly larger than those of the baselines ($p < 0.001$) with small to large effect size (see Table 10). Similarly, Table 10 also shows that our approach using LR classifier achieves a D2H value significantly lower than the baselines ($p < 0.01$) with small to large effect size. We provided the comparison results of our approach using other three classification techniques in the Appendix (see Section A6). These results indicate that our approach can predict the future SP changes better than the baselines. Moreover, the results also show that using various metrics to predict future SP changes achieve the accuracy better than considering a single metric (like OneR) or using only text description (like LDA-based and text-based approaches), suggesting that other factors than the text may also play a role in predicting the SP changes.

4.4.2 Influencing Metrics in the Classifiers

Our results in the previous section show that using various metrics can predict SP changes better than using a single metric. Hence, we examine our classifiers to better understand which metrics are influential in predicting the future SP changes. Figure 12 summarizes the ranks of the metrics in the seven studied projects. The x-axis indicates the metrics that appear as the five most influential metrics in our studied projects. The y-axis indicates the number of projects that have the metrics in that rank. We provide the metrics ranking results for each project in the Appendix (see Section A8).

Developer experience is the most influential metric for our datasets. Figure 12 shows that *reporter-stable-sp-rate* in the developer experience dimension is in the first rank (AS,

Table 10 The results of the one-sided Wilcoxon signed-rank tests and the effect size comparing between the performance of our Logistic Regression classifier (LR) and the baselines (i.e., LDA-based, Majority Vote, OneR, Text-based classifiers). Noted that Maj. stands for Majority Vote, and Text. stands for Text Classifiers

Project	AS				DM				ME				MU			
Measures	LR-OneR	LR-Maj.	LR-LDA	LR-Text.	LR-OneR	LR-Maj.	LR-LDA	LR-Text.	LR-OneR	LR-Maj.	LR-LDA	LR-Text.	LR-OneR	LR-Maj.	LR-LDA	LR-Text.
AUC↗	N/A	L***	L***	L***	N/A	L***	L***	L***	N/A	L***	L***	L***	N/A	L***	L***	L***
BACC↗	L***	M***	M***	L***	L***	L***	L***	L***	L***	L***	L***	L***	L***	L***	L***	L***
D2H↘	M***	M***	M***	M***	M***	L***	L***	L***	L***	L***	L***	L***	L***	L***	L***	L***
FAR↘	L°	L***	M***	L***	N*	M***	L°	L***	L°	M***	M***	S**	L°	M***	L°	L***
Recall↗	L***	S***	M***	N°	M***	N°	L***	L***	L***	M***	M***	M***	L***	M***	L***	L***

Project	TI				UG				XD			
Measures	LR-OneR	LR-Maj.	LR-LDA	LR-Text.	LR-OneR	LR-Maj.	LR-LDA	LR-Text.	LR-OneR	LR-Maj.	LR-LDA	LR-Text
AUC↗	N/A	L***	L***	L***	N/A	M***	M***	L***	N/A	L***	L***	L***
BACC↗	L***	L***	L***	L***	M***	S***	M***	M***	L***	L***	L***	L***
D2H↘	L***	M***	L***	L***	M***	S**	S***	M***	L***	L***	L***	L***
FAR↘	L°	L***	M***	L***	M°	M***	M***	N°	L°	L***	L°	L***
Recall↗	L***	N°	M***	L***	M***	N°	N°	M***	L***	M***	L***	S***

↗: Whether our approach achieve AUC, BACC, and Recall higher than the baselines

↘: Whether our approach achieve FAR and D2H lower than the baselines

Statistical significance: *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$, ° $p \geq 0.05$

Effect Size: L-Large ($r \geq 0.8$), M-Medium ($0.5 \leq r < 0.8$), S-Small ($0.2 \leq r < 0.5$),

N-Negligible ($r < 0.2$)

MU, TI, XD), the second rank (DM, ME), and the fourth rank (UG) of the most influential metrics in the studied projects. We also observe that on average, the likelihood that a work item will have SP changes decreases by 21%(UG)-80%(ME) when the value of *reporter-stable-sp-rate* increases from first quartile to third quartile. This observation indicates that *reporter-stable-sp-rate* has an inverse relationship with the likelihood that a work item will have SP changes after being assigned to a sprint.

Textual content is the second most influential metric for our datasets. Figure 12 shows that *text-gaussian* in the text dimension is in the first rank (DM, ME, MU), the second rank (TI, XD), and the fifth rank (UG) of the most influential metrics in the studied projects. These results indicate that textual content can be used to predict whether the work items will have SP changes.

We suspect that there might be specific keywords in the summary and description of work items that relate to the SP changes. Hence, we use the Local Interpretable Model Explanation (LIME) technique (Ribeiro et al. 2016) to determine the most influential keywords in the *text-gaussian* models that are constructed using the training dataset of the XD project. We find that "`stream`" is the keyword that is frequently identified as the most influential keyword in the XD dataset. We observe that `stream` is a module in Spring XD system[14]. We also observe that many work items that impact the `stream` module have SP changes. For example, XD-1280 and XD-2919 are the work items that impact the `stream` module and both of them had SP value increased from `10` to `20` and `3` to `5`, respectively. This observation suggests that an important keyword like a module name may be related to the SP changes.

Findings: Our Logistic Regression classifier (LR) can identify whether a work item will have SP changes after being assigned to a sprint with an average AUC of 0.69-0.8, an average Balanced Accuracy of 0.6-0.68, and an average Distance to Heaven of 0.33-0.47. We find that *reporter-stable-sp-rate* and *text-gaussian* are the most influential metrics in our LR classifier.
Implication: Our classifier can predict whether the SP will be changed after the work item is assigned to a sprint using the characteristics of work items. The past tendency of the work items reporter and the likelihood estimated based on textual content are the most important characteristics to anticipate the stability of SP.

5 Discussion

In this section, we discuss the implications of our case study results.

Our RQ1 shows that an average of 10% of the work items in the studied projects have SP changes after they were assigned to a sprint. Moreover, an average of 57% of these work items have their SP increased by 58%-100% relative to the SP values when they were assigned to the sprint. Nevertheless, 88% of them have SP changed only once. Our results

[14] https://docs.spring.io/spring-xd/docs/current/reference/html

suggest that although SP are less likely to be changed after the work item was assigned to a sprint, SP tend to be increased with a relatively large size change. Since a sprint is planned based on the SP estimates, SP changes (especially the increased ones) after the work items were assigned to a sprint may negatively impact the original sprint plan (Rubin 2012).

In RQ2, we find that the unchanged SP can better reflect the development time than the changed SP_{sprint} and the changed SP_{last}. This finding indicates that the stable (unchanged) SP can better reflect the development time than the unstable (changed) SP, suggesting that aiming for stable SP estimation will also lead to more accurate SP and the corresponding sprint plan. In other words, to achieve a more accurate SP estimation, the SP should be refined before or during sprint planning.

Our RQ3 shows that the work items with changed SP tend to have information changes after being assigned to a sprint more often than those with unchanged SP. Our findings are consistent with the Grounded Theory study of (Hoda and Murugesan 2016) and (Bick et al. 2018) who reported that the practitioners may re-estimate the effort when the new information becomes available. However, we find that not all types of information changes are associated with the SP changes. Our RQ3 shows that the information changes for updating the scope of work are often found with the work items with changed SP, while the information changes for clarification are often found with the work items with unchanged SP. Our results suggest that to acquire confidence that the sprint plan is created based on the latest and correct information, the scope of work and SP should be revisited before (or during) sprint planning.

Although our RQ2 and RQ3 suggest that SP and scope of work should be revisited before or during sprint planning, it could still be tedious for the team to anticipate the SP changes or ensure the information for all the work items within a limited time. Hence, our classifier developed in RQ4 should help the team focuses on the work items that are likely to have SP changes in the future. Using a logistic regression technique and the 41 metrics, our LR classifier can predict whether a work item will have SP changes with an AUC of 0.69-0.8, which are better than the baseline approaches. These results suggest that our classifier can predict whether a work item will have SP changes after being assigned to a sprint. Hence, our approach should help the team better manage and prepare for the unreliability in SP estimation. For example, instead of revisiting all the work items and their SP estimates in the product backlog (or sprint backlog), the team can review a small set of work items that are predicted to have SP changes after being assigned to a sprint. In addition, the team can reserve (or adjust) the amount of buffer time during the sprint for the possible changes of those work items (Rubin 2012).

Our RQ4 also shows that the *developer experience* and *textual content* are the most influential metrics in our LR classifier. These two metrics are also reported in the prior studies to be useful in predicting SP using machine learning models and deep learning models (Porru et al. 2016; Scott and Pfahl 2018; Choetkiertikul et al. 2019). Furthermore, we observed that the experienced developers tend to provide stable SP estimation in all seven studied projects. This suggests that having an experienced developer involved in SP estimation or sprint planning may decrease the likelihood that SP will be changed in the future. We also observed that some keywords in the summary and description of work items (e.g., module name) could indicate SP changes. Nonetheless, it is still unclear what part of the summary and description can determine the SP changes. Future work should investigate what keywords or certain information that can indicate the SP changes.

6 Threats to Validity

We now discuss the possible threats to the validity of our study.

6.1 External Validity

Our results may not be generalized to all kinds of software projects. The process and practices may be varied from project to project (Masood et al. 2022). In this paper, we perform a large-scale empirical study on 19,349 work items from seven open-source projects that we carefully select based on the project selection criteria (see Section 3.1). We find that our studied results are generally similar across the seven studied projects. These projects actively use SP and are various in size (i.e., 19-245 developers), active lifespan (i.e., 5-8 years), and developers community. Nevertheless, the findings of our work may need to be revisited for other software projects with different characteristics.

6.2 Internal Validity

Our study is conducted solely based on the data that is recorded in JIRA. However, SP changes and information changes may be recorded in other repositories that are related to a work item, e.g., project documentation. Linking information across repositories may allow us to access additional data for SP changes and information changes. However, linking multiple repositories is not a trivial task (Vargas et al. 2018; Tiwari et al. 2016). Nonetheless, we carefully check the project documentation of the studied projects[15][16][17][18][19] and find that JIRA is their main repository to store the information of a work item and the SP estimation. Hence, we believe that the results of our study are less likely impacted.

Since the time point of the sprint planning is not available in the data, we use the time point when the work items were assigned to a sprint as a reference point of sprint planning. This is because the work items are normally assigned to the sprint during sprint planning (Schwaber and Sutherland 2020). However, the work items may be pre-assigned to the sprint prior to sprint planning. To address this possible threat, we confirm with the core contributors of the studied projects about their sprint planning practice. They informed us that the team typically assign the work items to the sprint during sprint planning (see Section 4). Hence, it is less likely that the work items are pre-assigned before sprint planning.

Our types of information changes in RQ3 are derived based on the open coding process performed by the first and second authors. The subjective facets of the coders (e.g., experiences or expertise) can influence the results of the open coding process and subsequent analysis. To mitigate this threat, we validate our coded types with two additional coders and assess the inter-rater reliability (Syed and Nelson 2015; McHugh 2012). The result shows a strong agreement on the types of information changes, which indicates that our results are not subjective to the first and second authors.

[15] https://mesos.apache.org/documentation/latest/reporting-an-issue
[16] https://github.com/mulesoft/mule/blob/master/CONTRIBUTING.md
[17] https://github.com/spring-projects/spring-xd
[18] https://wiki.appcelerator.org/display/guides2/Contributing+to+Titanium
[19] https://developer.lsst.io/work/flow.html

6.3 Construct Validity

We study the projects that actively use SP as an effort unit. However, other effort units (e.g., man-hours) may also be used. To mitigate this threat, we asked the core contributors to confirm that they used SP. They informed us that they estimate SP during sprint planning. In addition, we manually checked other properties of work items and did not find any other effort units are being used. Hence, it is unlikely that these projects used other effort units.

We extract SP and their changes recorded in JIRA work item. However, the prevalence of SP changes might be inflated, e.g., the initial SP are a temporary value. To mitigate this threat, we minimize the noise in the dataset by selecting the work items with a reasonable range of SP (see Section 3.1, DP-3).

Since the developers may perform documentation after sprint planning, we only consider a work item as having changed SP if there is an SP change event occurred later than one hour after the work item was assigned to a sprint. Yet, the one hour threshold may not be the best cutoff for this purpose. Hence, we examine the proportion of SP changes that occurred in each hour after the work items were assigned to a sprint. We find that on average, 21% of work items had SP changes within the first hour after they were assigned to a sprint, while 1%-4% of work items had SP changes in the subsequent hours. We provide the charts that show this distribution in the Appendix (Section A2). This observation suggests that the SP changes in the first hour after sprint planning are common and these may be related to documentation activities after sprint planning. Hence, the SP changes and information changes within this period are likely to be acceptable.

When selecting the studied projects, we contact core contributors to confirm that their projects actively use SP and allow SP to be changed according to Criteria 2 and 3 (see Section 3.1). It is possible that they may not fully recall the policy or norm when they respond to our email. To mitigate the risk of inaccurate information, for each project, we contacted 2-10 core contributors to cross-check their answers. In total, we sent emails to 47 core contributors and 12 core contributors provided the information of the relevant projects. Furthermore, we also check if the core contributors actively work on the projects. To do so, we check the number of code commits of these core contributors. We found that all the core contributors are the top ten percentile most frequent committers of their projects. In addition, The core contributors from DM and XD are the authors of the project management guide.[20][21] Hence, we believed that all the core contributors have a sufficient understanding of the project management activities and provide us the accurate results.

We extract the development time from the time when the status of a work item was set as '*in progress*' until it was set as '*done*', '*completed*', or '*fixed*'. However, the extracted development time may not precisely represent the time spent in development as it may also include other activities, e.g., communication overhead or idle time. We also observed that few work items were re-opened multiple times. Hence, developers may work on those work items in several periods. To mitigate this threat, we tried to exclude the time spent on other activities by using an approach of (Choetkiertikul et al. 2019). In particular, we aggregate (sum) the time interval that the work items were under the *in progress* status instead of relying on only one interval.

[20] https://dmtn-020.lsst.io

[21] https://docs.spring.io/spring-xd/docs/current-SNAPSHOT/reference/html

In RQ4, we measure the performance of the classifier using the threshold-dependent metrics, i.e., Balanced Accuracy, Distance to Heaven, False Alarm Rate, and Recall. To do so, we calculate the optimal threshold based on the training dataset using our approach in Section 3.3.4.2 (EC-1). Hence, the performance of the classifier may be varied when applying different thresholds. Nonetheless, a prior study suggests that the threshold-independent metric (i.e., AUC) should be used as it is less sensitive than the threshold-dependent metrics (Tantithamthavorn et al. 2018). In addition, our RQ4 shows that our classifier achieves AUC significantly better than the baselines with small to large effect size.

We build the classifiers in RQ4 based on four machine learning techniques (i.e., LR, RF, SVM, and CART). Other classification techniques (e.g., neural network techniques) may achieve a different prediction accuracy. However, we opted to use these four classification techniques because they can provide a set of influential metrics for the subsequent analyses of RQ4 (see Section 3.3.4.3), which neural network techniques cannot.

In RQ4 (CC-2), we remove the highly-correlated metrics using AutoSpearman (Jiarpakdee et al. 2018). However, other techniques might provide a different set of metrics. Nevertheless, for a sake of reproducibility, we opt to use AutoSpearman which has been shown that it can provide a consistent set of metrics.

7 Related Work

In this section, we discuss the related work with respect to effort estimation and automated approaches for effort estimation.

7.1 Effort Estimation

Prior studies investigated the effort estimation accuracy against the development time. (Kirmani and Wahid 2015) investigated whether the Use Case Points can reflect the development time (in hours) spent in a project. (Choetkiertikul et al. 2019) found that the SP of a work item can reflect the development time. However, these two studies focused on the latest value of the estimated effort. Indeed, the estimated effort may be changed even after sprint planning. Yet, it is unclear whether the changed or unchanged SP are more accurate in terms of reflecting the development time. In this paper, our RQ2 investigate the ability to reflect the development time of unchanged SP, changed SP_{sprint}, and changed SP_{last}. The results show that the unchanged SP can better reflect the development time than changed SP_{sprint} and changed SP_{last}.

Prior studies suggested that the quality of the available information plays a major role in estimation accuracy. Based on controlled experiments, (Jørgensen and Gruschke 2009) found that unclear task specification is one of the reasons that lead to an inaccurate estimation. Based on a survey, (Britto et al. 2015) found that unclear, unstable, and misdocumented information leads the teams to overlook the important development activities when estimating effort. Nevertheless, these prior studies have not yet investigated whether the association between SP changes and information changes are statistically significant. In this paper, our statistical analysis in RQ3 has confirmed that the SP changes are significantly associated with the information changes.

Prior studies pointed out that information changes could possibly cause SP changes. (Hoda and Murugesan 2016) conducted a Grounded Theory study and (Bick et al. 2018) conducted a case study with Agile practitioners. Both studies reported that the Agile practitioners may re-estimate the work item when the related information is changed. However, a recent survey study pointed out that practitioners would re-estimate a work item if the information change is significant (Pasuksmit et al. 2021). Yet, it is still unclear what types of information change are associated with SP changes. In this work, we find that work items with changed SP tend to also have information changes for changing scope more than the other types, e.g., clarification.

7.2 Approaches for Effort Estimation

Since effort estimation is time-consuming (Hoda and Murugesan 2016), prior studies proposed automated approaches to predict the effort at the project level (Manga and Blamah 2014; Kuan 2017; Sarro and Petrozziello 2018; Sarro et al. 2016; Tawosi et al. 2021) and the sprint level (Choetkiertikul et al. 2017; Ramessur and Nagowah 2021). Recent approaches were proposed to predict the effort at the work item level (Porru et al. 2016; Scott and Pfahl 2018; Choetkiertikul et al. 2019; Alhamed and Storer 2021; Fu and Tantithamthavorn 2022; Tawosi et al. 2022). Although these approaches could help practitioners estimate and update the effort anytime, the change of effort estimation after sprint planning could invalidate the sprint plan. Hence, our proposed classifier would help practitioners know in advance whether the estimated SP are likely to be changed after sprint planning, allowing them to gain more confidence in using the SP to plan the sprint.

Recent studies developed prediction models to support the effort estimation practices. For example, (Basri et al. 2016) and (Shah et al. 2018) proposed the models to predict the effort required to address requirement changes based on the change impact analysis and effort estimation techniques. (Dehghan et al. 2017) developed a model to predict whether a requirement will be completed within a development iteration. (Choetkiertikul et al. 2017) proposed an approach to predict the delay to deliver a work item based on the risk factors extracted from the work items properties. However, none of these approaches aimed to support the practitioners to prepare for uncertainty in effort estimation at the work items level. In this study, we aimed to fulfill this gap with our automated approach that aim to help the team better prepare for the unreliability in SP estimation.

Recent studies proposed manual techniques to help the teams achieve a reliable SP estimation. For example, (Popli and Chauahn 2015) suggested to aggregate the SP of subtasks when estimating SP of the associated work item. (Usman et al. 2018) developed a personalized checklist for a software company to help the estimators improve the estimation stability by recalling relevant factors. However, these manual techniques require the additional effort of practitioners, which can be a significant overhead (Usman et al. 2018). In contrast to the prior studies, we proposed an automated approach to predict whether the SP will be changed after a work item is assigned to a sprint using the available information in the work item. Using our approach, the teams can prepare for or be aware of the anticipated SP changes with minimal effort. For example, instead of reviewing all work items in the sprint, the team can review the related information of the predicted work items before sprint planning. The team can also include an expert in effort estimation or reserve more buffer time in the sprint for the anticipated changes (Rubin 2012).

8 Conclusions

Story Points (SP) estimation helps an Agile team to create a sprint plan. While SP can be changed to maintain the ability to reflect the relative effort of the work item, SP changes after sprint planning may invalidate the sprint plan. Yet, the nature of SP changes is still unclear. Hence, we conduct a mixed-methods empirical study and develop an approach to address the following research questions:

(RQ1) To what degree do Story Points change?
(RQ2) Do changed Story Points better reflect the development time than the ones that remain unchanged?
(RQ3) To what extent are Story Points changes are associated with information changes?
(RQ4) Can we predict whether a work item will have Story Points changes?

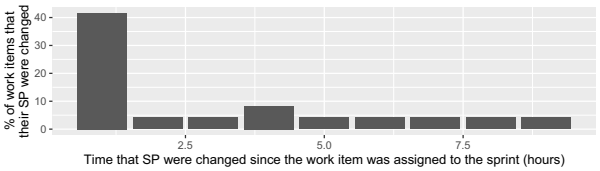
Our results show that although the SP were not frequently changed after the work items were assigned to a sprint (an average of 10%), they tend to be increased with a relatively large size change (i.e., 58%-100% relative to the SP value when they were assigned to the sprint). We also find that the stable (unchanged) SP can better reflect the development time than the re-estimated (changed) SP. Our qualitative result shows that the SP tend to be changed corresponding to the change of scope of work. Our proposed approach can predict whether the SP will be changed after the work item is assigned to a sprint. Our results suggest that to achieve more stable SP estimation and sprint plan, the scope of a work item should be revisited before sprint planning to make sure that the recorded information and SP are up to date. Instead of spending effort on revisiting all the work items in the product backlog or sprint backlog, our classifier should help the team focus on a subset of work items that are likely to have SP changes after being assigned to a sprint.

Appendix

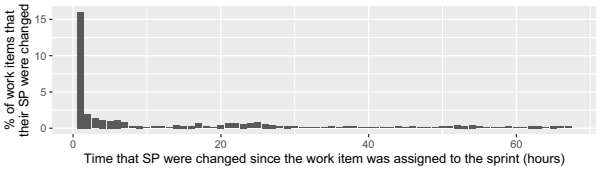
To improve the understandability and reproducibility of this study, we provide additional illustrations and explain the technical details of our approaches and results below.

Appendix A - Threshold for Acceptable Changes (Additional illustration)

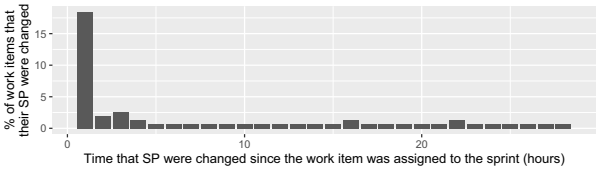
In this study, we only consider the SP changes that occurred later than one hour after the work items were assigned to the sprint. This is because we find that there are a large proportion of SP changes occurred within the one hour gap. In particular, on average, 21% of work items with changed SP had SP changes within the first hour after they were assigned to the sprint, while 1%-4% of work items with changed SP had SP changes in each of the subsequent hours. Figure 13a–13g shows the number of work items that their SP were changed within each hour after they were assigned to a sprint.



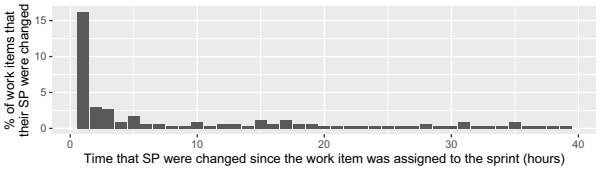
(a) AS



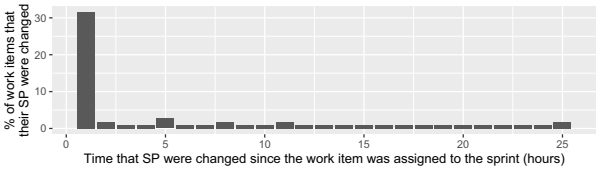
(b) DM



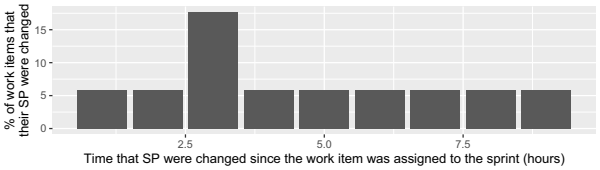
(c) ME



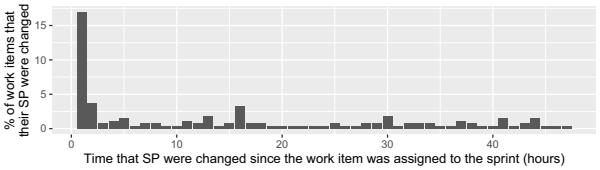
(d) MU



(e) TI



(f) UG



(g) XD

Fig. 13 The number of work items that their SP were changed within each hour after they were assigned to a sprint

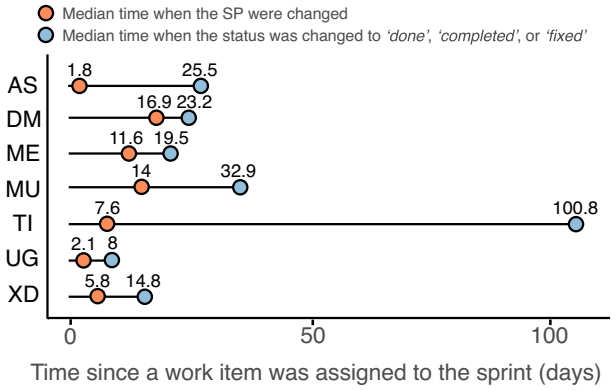


Fig. 14 An average (median) length of time when the SP were changed after a work item was assigned to a sprint and the work item was marked as *done*, *completed*, or *fixed*

Appendix B - RQ1 (Additional illustration)

In RQ1, we find that the SP of the work items with changed SP were typically changed 1.8(AS)-16.9(DM) days after they were assigned to a sprint, while these work items took 14.8(XD)-100.8(TI) days to arrive at the close status. This finding suggests that a work item typically has SP changes few days after it was assigned to a sprint. Figure 14 shows an average (median) length of time when the SP were changed after a work item was assigned to a sprint and the time when the work item was marked as *done*, *completed*, or *fixed*.

Appendix C - RQ2 (Analysis approach)

To address our RQ2, we construct Linear Mixed-Effects models to assess the relationship between SP and development time while considering the type of work item as a random effect. We construct the Linear Mixed-Effects models using the `lme` function from R nlme package (Pinheiro et al. 2019).

After that, we assess the goodness of fit of the models and examine the relationship between SP and development time. To assess the goodness of fit of the models, we use a Log-Likelihood Ratio (LR) test to compare our models with the null models (i.e., $developmenttime \sim 1$). We use the `lrtest` function from the R lmtest package (Hothorn et al. 2019) to compute LR χ^2 and its significance level (p-value). We then examine our models to assess the relationship between the SP and development time using Wald χ^2 statistics. We use the *Anova Type-III* function of the R car package (Fox and Weisberg 2019) to obtain the Wald χ^2 values.

Appendix D - RQ3 (Quantitative analysis)

In our quantitative analysis study, we examine the association between the number of work items with changed SP and the number of work items with changed information. Figure 15 provides an illustrative example of our approach to identify the work items with information

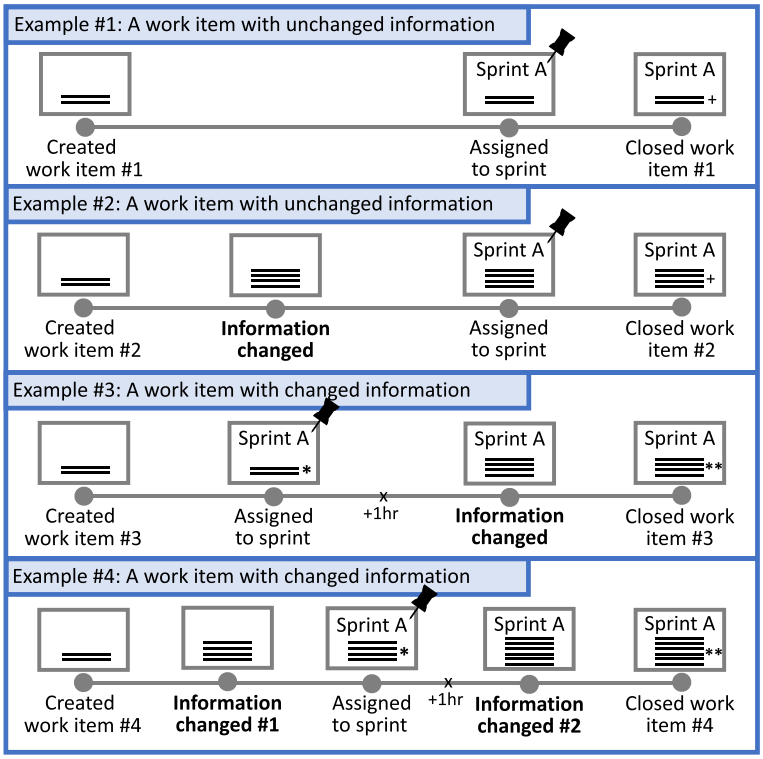


Fig. 15 Examples of the identification of information changes based on the activity log of a work item. + : Unchanged information, * : information$_{@\text{sprint}}$, ** : information$_{@\text{close}}$

change. The work items #1 and #2 in the figure have no information change event after being assigned to a sprint. Hence, we identify them as the work items with unchanged information. On the other hand, work items #3 and #4 have an information change event that occurs later than one hour after being assigned to a sprint. Therefore, we identify them as the work items with changed information. Then, we use the one-sided Fisher's Exact test to test whether the work items with changed SP have changed information more often than the work items with unchanged SP. We use the `fisher.test` function of the R stats package (R Core Team 2013) to determine the odds ratios and their statistical significance (p-value).

Appendix E - RQ4 (Constructing and Evaluating the Classifiers)

(CC-1) Extracting Characteristics

In our RQ4, we construct the classifier to predict whether a work item will have SP changes after being assigned to a sprint. To do so, we first characterize the information provided in

the work items using 41 metrics which are grouped into six dimensions, i.e., (1) activity, (2) collaboration, (3) completeness, (4) experience, (5) readability, and (6) Text.

Collaboration Dimension. To extract the *Collaboration* dimension, we use Python NetworkX package[22] to construct the collaboration network of a reporter, i.e., *collab-in-degree*, *collab-out-degree*, *collab-total-degree*, *collab-kcoreness*, *collab-clustering-coefficient*, *collab-closeness-centrality*, *collab-betweenness-centrality*, *collab-eigenvector-centrality*.

Text Dimension. To estimate the textual content score of the work items for the *Text* dimension, we build the models using four Naive-Bayes algorithms. More specifically, we use the `MultinomialNB`, `GaussianNB`, `BernoulliNB`, and `ComplementNB` functions of Scikit-learn Python package (Pedregosa et al. 2011) to build Multinomial Naive-Bayes (McCallum and Nigam 1998), Bernoulli Naive-Bayes (McCallum and Nigam 1998), Gaussian Naive-Bayes (Geiger and Heckerman 1994), and Complement Naive-Bayes (Rennie et al. 2003) models, respectively.

(EC-2) Baseline Approaches

In this study, we compare the performance of our classifiers against One Rule classification algorithm (OneR (von Jouanne-Diedrich 2017)), Latent Dirichlet Allocation (LDA (Blei et al. 2003)), and Majority Vote. We use `OneR` function of the R OneR Package to construct OneR classifier (von Jouanne-Diedrich 2017). We use `LdaModel` function of the gensim Python Package to extract the topic from the summary and description of the work items (Rehurek et al. 2011). For Majority Vote approach, we determine an optimal cutpoint for each metric to vote whether a work item will have SP changes. Then, the approach will give the final decision based on the majority vote of all metrics. We use `cutpointr` function of the cutpointr R package to determine the optimal cutpoint for each metric (Thiele and Hirschfeld 2021). Then, we compare the performance distributions of our classifiers against the performance distributions of baselines using a one-sided exact Wilcoxon signed-rank test (Wilcoxon 1992). To do so, we use the `wilcoxsign_test` function of R coin package (Hothorn et al. 2006). Table 11 shows the performance improvement of our classifier over the three baselines.

10.5.3 Examining the Influence of Metrics

After we constructed our classifiers, we examine the influence of the metrics in our classifiers. We find that *text-gaussian* (i.e., the likelihood that a work item will have SP changes based on textual content) is the second most influence metric in our LR classifier. Then, we use the Local Interpretable Model Explanation (LIME) technique to determine the influential keywords in the classifiers using the `explain_instance` function from LIME Python package (Ribeiro et al. 2016).

[22] https://networkx.github.io

Table 11 The results of the one-sided Wilcoxon signed-rank tests and the effect size comparing between the performance of our approach (using LR, RF, SVM, and CART) and the baselines (i.e., LDA-based, Majority Vote, OneR, Text-based classifiers). Noted that Maj. stands for Majority Vote, and Text. stands for Text Classifiers

		Effect Size														
Proj.	Measure	LR-OneR	LR-Maj.	LR-LDA	LR-Text.	RF-OneR	RF-Maj.	RF-LDA	RF-Text.	SVM-OneR	SVM-Maj.	SVM-LDA	SVM-Text.	CART-OneR	CART-Maj.	CART-LDA	CART-Text.
AS	AUC↗	N/A	L***	L***	L***	N/A	M***	M***	L***	N/A	L***	L***	L***	N/A	S*	S***	M***
	BACC↗	L***	M***	M***	L***	S°	M°	M°	M°	M***	S***	S***	M***	S**	N°	N°	N°
	D2H↘	M***	M***	M***	M***	S°	M°	S°	M°	N°	N°	N°	S°	N°	N°	N°	N°
	FAR↘	L°	L***	M***	L***	N°	L***	L***	L***	L°	L***	L***	L***	M°	L***	M***	L***
	Recall↗	L***	S***	M***	N°	S°	M°	M°	L°	M***	S°	N°	M°	S**	S°	N°	S°
DM	AUC↗	N/A	L***	L***	L***	N/A	L***	L***	L***	N/A	L***	L***	L***	N/A	M***	L***	L***
	BACC↗	L***	L***	L***	L***	S***	N°	L***	L***	L***	L***	L***	L***	M***	M***	L***	L***
	D2H↘	M***	L***	L***	L***	M°	L°	L***	L°	M***	L***	L***	L***	M***	S***	L***	M***
	FAR↘	N*	M***	L°	L***	L***	L***	L***	L***	S***	L***	L°	L***	N*	M***	L°	L***
	Recall↗	M***	N°	L***	L***	L°	L°	M***	L°	M***	S°	L***	M***	S***	S°	L***	N°
ME	AUC↗	N/A	L***	L***	L***	N/A	L***	L***	L***	N/A	L***	L***	L***	N/A	S***	M***	M***
	BACC↗	L***	L***	L***	L***	S***	N°	M***	M***	L***	L***	L***	L***	M***	M***	M***	M***
	D2H↘	L***	L***	L***	L***	N°	L°	L°	M°	L***	S***	M***	M***	M***	N°	S**	S***
	FAR↘	L°	M***	M***	S**	L***	L***	L***	L***	L°	L***	L***	M***	M°	L***	L***	M***
	Recall↗	L***	M***	M***	M***	N°	L°	L°	L°	L***	S°	N*	S***	M***	S°	N°	N*
MU	AUC↗	N/A	L***	L***	L***	N/A	L***	L***	L***	N/A	L***	L***	L***	N/A	S**	L***	M***
	BACC↗	L***	L***	L***	L***	M***	S°	L***	S***	L***	L***	L***	L***	M***	M***	M***	M***
	D2H↘	L***	L***	L***	L***	M***	L°	S°	L°	L***	M***	L***	L***	L***	S**	L***	S***
	FAR↘	L°	M***	L°	L***	N°	L***	L***	L***	L°	L***	M°	L***	L°	M***	M°	M***
	Recall↗	L***	M***	L***	L***	M***	L°	M°	L°	L***	N°	L***	M***	L***	N°	L***	N°

Table 11 (continued)

Proj.	Measure	Effect Size															
		LR-OneR	LR-Maj.	LR-LDA	LR-Text.	RF-OneR	RF-Maj.	RF-LDA	RF-Text.	SVM-OneR	SVM-Maj.	SVM-LDA	SVM-Text.	CART-OneR	CART-Maj.	CART-LDA	CART-Text.
TI	AUC↗	N/A	L***	L***	L***	N/A	L***	L***	L***	N/A	L***	L***	L***	N/A	N°	L***	M***
	BACC↗	L***	L***	L***	L***	M***	S°	L***	M***	L***	L***	L***	L***	M***	S***	L***	L***
	D2H↘	L***	M***	L***	L***	M***	L°	M°	M°	L***	N°	L***	L***	M***	M°	S*	N*
	FAR↘	L°	L***	M***	L***	L***	L***	L***	L***	L°	L***	L***	L***	M°	L***	L***	L***
	Recall↗	L***	N°	M***	L***	M***	L°	M°	L°	L***	M°	M***	M***	M***	L°	N°	N°
UG	AUC↗	N/A	M***	M***	L***	N/A	M***	M***	L***	N/A	M***	M***	L***	N/A	N°	S***	M***
	BACC↗	M***	S***	M***	M***	N°	N°	S*	S***	S***	S**	M***	M***	S**	N°	S***	M***
	D2H↘	M***	S**	S***	M***	S°	M°	M°	N°	S***	S°	N°	S***	S**	S°	N°	S***
	FAR↘	M°	M***	M***	N°	M***	L***	L***	M***	N°	L***	L***	S***	N°	M***	M***	S***
	Recall↗	M***	N°	N°	M***	S°	L°	M°	S°	S***	S°	S°	S***	N*	S°	S°	S*
XD	AUC↗	N/A	L***	L***	L***	N/A	L***	L***	L***	N/A	L***	L***	L***	N/A	S***	L***	S***
	BACC↗	L***	L***	L***	L***	L***	S**	L***	S***	L***	L***	L***	L***	L***	M***	L***	M***
	D2H↘	L***	L***	L***	L***	M***	L°	L***	L°	L***	M***	L***	M***	L***	N°	L***	N°
	FAR↘	L°	L***	L°	L***	M***	L***	L***	L***	M°	L***	L°	L***	S°	L***	M°	L***
	Recall↗	L***	M***	L***	S***	M***	L°	L***	L°	L***	N°	L***	N°	L***	S°	L***	M°

↗: Whether our approach achieve AUC, BACC, and Recall higher than the baselines

↘: Whether our approach achieve FAR and D2H lower than the baselines

Statistical significance: *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$, ° $p \geq 0.05$

Effect Size: L-Large $r < 0.8$, M-Medium $0.5 \leq r < 0.8$, S-Small $0.2 \leq r < 0.5$, N-Negligible $\leq r < 0.2$

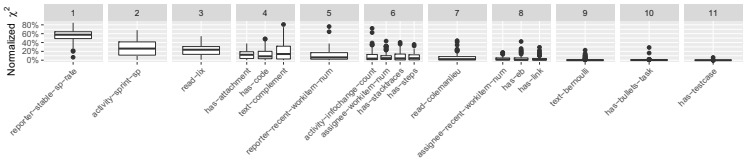
Appendix F - URL of JIRA work items

This section provides the URLs of the work items mentioned in the study.

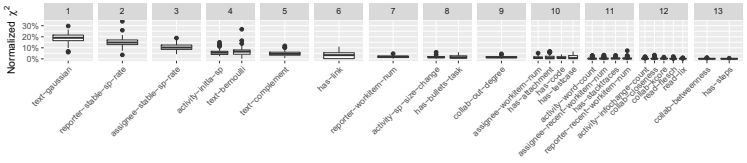
DM-814: https://jira.lsstcorp.org/browse/DM-814
DM-2683: https://jira.lsstcorp.org/browse/DM-2683
DM-7267: https://jira.lsstcorp.org/browse/DM-7267
DM-9282: https://jira.lsstcorp.org/browse/DM-9282
DM-8007: https://jira.lsstcorp.org/browse/DM-8007
DM-12211: https://jira.lsstcorp.org/browse/DM-12211
DM-14172: https://jira.lsstcorp.org/browse/DM-14172
DM-13489: https://jira.lsstcorp.org/browse/DM-13489
MU-15159: https://www.mulesoft.org/jira/browse/MULE-15159
TI-2568: https://jira.appcelerator.org/browse/TIMOB-2568
TI-19356: https://jira.appcelerator.org/browse/TIMOB-19356
TI-23156: https://jira.appcelerator.org/browse/TIMOB-23156
XD-1280: https://jira.spring.io/browse/XD-1280
XD-1503: https://jira.spring.io/browse/XD-1503
XD-2919: https://jira.spring.io/browse/XD-2919

Appendix G - The metrics influence ranks in the LR classifiers

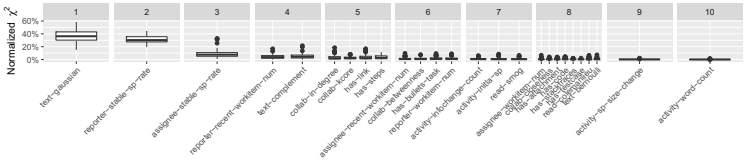
Figure 16a–g show the rank of the metrics by their influence to the prediction of LR classifiers.



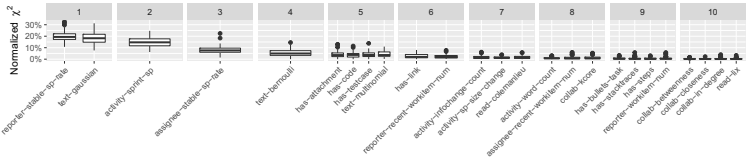
(a) AS



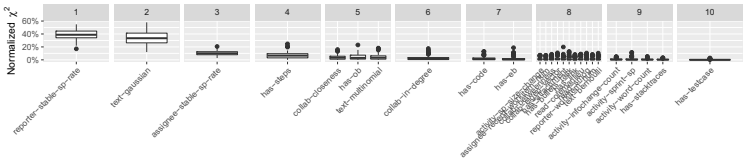
(b) DM



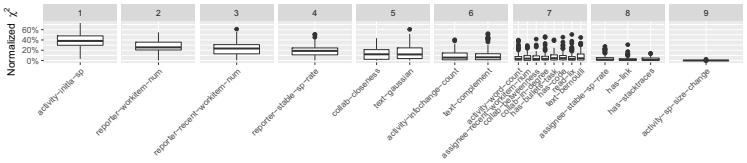
(c) ME



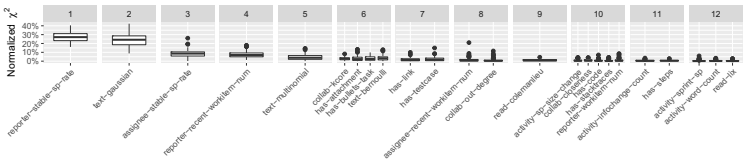
(d) MU



(e) TI



(f) UG



(g) XD

Fig. 16 The rank of the metrics by their influence to the prediction of LR classifiers

Acknowledgements We thank Associate Professor Rashina Hoda from Monash University, Australia (one of the experts in Agile research) for her constructive feedback that substantially improves our work. We also would like to thank the external coder who voluntarily helped us validate our open-coding outcome.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions. P. Thongtanunam was partially supported by the Australian Research Council's Discovery Early Career Researcher Award (DECRA) funding scheme (DE210101091).

Declarations

Conflicts of interests/Competing interests The authors have no relevant conflicts of interests, competing interests, financial interests, or non-financial interests to disclose.

References

Agrawal A, Fu W, Chen D, Shen X, Menzies T (2019) How to "DODGE" Complex Software Analytics. TSE

Alhamed M, Storer T (2021) Playing Planning Poker in Crowds: Human Computation of Software Effort Estimates. In: Proceedings of the ICSE, pp 1–12

Basri S, Kama N, Haneem F, Ismail SA (2016) Predicting effort for requirement changes during software development. In: Procedings of the SoICT, pp 380–387

Bick S, Spohrer K, Hoda R, Scheerer A, Heinzl A (2018) Coordination Challenges in Large-Scale Software Development: A Case Study of Planning Misalignment in Hybrid Settings. TSE 44(10):932–950

Blei DM, Ng AY, Jordan MI (2003) Latent Dirichlet Allocation. JMLR 3:993–1022

Britto R, Mendes E, Börstler J (2015) An Empirical Investigation on Effort Estimation in Agile Global Software Development. In: Proceedings of the ICGSE, pp 38–45

Cerpa N, Bardeen M, Kitchenham B, Verner J (2010) Evaluating logistic regression models to estimate software project outcomes. IST 52(9):934–944

Chaparro O, Lu J, Zampetti F, Moreno L, Di Penta M, Marcus A, Bavota G, Ng V (2017) Detecting Missing Information in Bug Descriptions. In: Proceedings of the ESEC/FSE, pp 396–407

Charmaz K (2014) Constructing grounded theory

Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) Smote: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357

Choetkiertikul M, Dam HK, Tran T, Ghose A (2017) Predicting the delay of issues with due dates in software projects. EMSE 22(3):1223–1263

Choetkiertikul M, Dam HK, Tran T, Ghose A, Grundy J (2017) Predicting delivery capability in iterative software development. TSE 44(6):551–573

Choetkiertikul M, Dam HK, Tran T, Pham TTM, Ghose A, Menzies T (2019) A deep learning model for estimating story points. TSE 45(07):637–656

Coelho E, Basu A (2012) Effort Estimation in Agile Software Development using Story Points. IJAIS 3(7):7–10

Cohen J (2013) Statistical power analysis for the behavioral sciences

Cohn M (2006) Agile estimating and planning

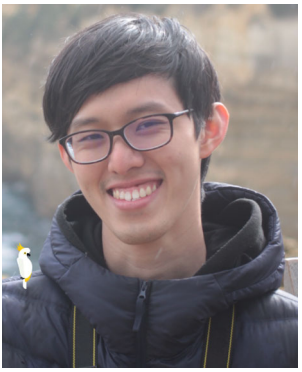
Coleman M, Liau TL (1975) A Computer Readability Formula Designed for Machine Scoring. J Appl Psychol 60(2):283–284

Corazza A, Di Martino S, Ferrucci F, Gravino C, Sarro F, Mendes E (2013) Using tabu search to configure support vector regression for effort estimation. EMSE 18(3):506–546

Dam HK, Tran T, Grundy J, Ghose A, Kamei Y (2019) Towards effective AI-powered agile project management. In: ICSE-NIER, pp 41–44
Dehghan A, Neal A, Blincoe K, Linaker J, Damian D (2017) Predicting Likelihood of Requirement Implementation within the Planned Iteration: An Empirical Study at IBM. In: Proceedings of the MSR, pp 124–134
Efron B (1983) Estimating the error rate of a prediction rule: improvement on cross-validation. JASA 78(382):316–331
El Zanaty F, Hirao T, McIntosh S, Ihara A, Matsumoto K (2018) An Empirical Study of Design Discussions in Code Review. In: Proceedings of the ESEM, pp 11:1–11:10
Fan Y, Xia X, Lo D, Hassan AE (2018) Chaff from the Wheat: Characterizing and Determining Valid Bug Reports. TSE, pp 495–525
Fisher RA (1992) Statistical Methods for Research Workers
Flesch R (1948) A New Readability Yardstick. J Appl Psychol 32(3):221–233
Fowler FM (2019) Navigating hybrid scrum environments
Fox J, Weisberg S (2019) An r companion to applied regression, Third. Sage, Thousand Oaks CA
Fu M, Tantithamthavorn C (2022) GPT2SP: A Transformer-Based Agile Story Point Estimation Approach. TSE. To Appear
Geiger D, Heckerman D (1994) Learning gaussian networks. In: Uncertainty Proceedings, pp 235–243
Gralha C, Damian D, Wasserman A, Goulão M, Araújo J (2018) The evolution of requirements practices in software startups. In: Proceedings of the ICSE, pp 823–833
Grenning J (2002) Planning Poker or How to avoid analysis paralysis while release planning, Hawthorn Woods: Renaissance Software Consulting. vol 3, pp 22–23
Gunning R (1952) The Technique of Clear Writing
Hanley JA, McNeil BJ (1982) The Meaning and Use of the Area Under a Receiver Operating Characteristic (ROC) Curve. Radiology 143(1):29–36
Harrell Jr FE (2015) Regression Modeling Strategies: with Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis
Harrell Jr. FE (2019) Package 'rms'. https://CRAN.R-project.org/package=rms, R package version 5.1-4
Haugen NC (2006) An Empirical Study of Using Planning Poker for User Story Estimation. In: AGILE, pp 9–34
Hoda R, Murugesan LK (2016) Multi-Level Agile Project Management Challenges: A Self-Organizing Team Perspective. JSS 117:245–257
Holte RC (1993) Very Simple Classification Rules Perform Well on Most Commonly Used Datasets. Mach Learn 11(1):63–90
Hothorn T, Hornik K, Van De Wiel MA, Zeileis A (2006) A Lego system for conditional inference. The American Statistician 60(3):257–263
Hothorn T, Zeileis A, Farebrother RW, Cummins C, Millo G, Mitchell D, Zeileis MA (2019) Package 'lmtest'. Testing linear regression models. https://cran.r-project.org/web/packages/lmtest/lmtest.pdf
Jiarpakdee J, Tantithamthavorn C, Treude C (2018) AutoSpearman: Automatically Mitigating Correlated Software Metrics for Interpreting Defect Models. In: Proceedings of the ICSME, pp 92–103
Jonathan Anderson (1983) Lix and Rix: Variations on a Little-known Readability Index. J Read 26(6):490–496
Jørgensen M, Gruschke TM (2009) The impact of lessons-learned sessions on effort estimation and uncertainty assessments. TSE 35(3):368–383
Kincaid JP, Fishburne Jr RP, Rogers RL, Chissom BS (1975) Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel
Kirmani MM, Wahid A (2015) Revised use case point (re-ucp) model for software effort estimation. Int J Adv Comput Sci Appl 6(3):65–71
Kocaguneli E, Menzies T, Keung JW (2011) On the Value of Ensemble Effort Estimation. TSE 38(6):1403–1416
Kruskal WH, Wallis WA (1952) Use of ranks in one-criterion variance analysis. JASA 47(260):583–621
Kuan S (2017) Factors on software effort estimation. IJSEA 8(1):23–32
Liaw A, Wiener M (2018) Package 'randomForest': Breiman and Cutler's random forests for classification and regression. https://CRAN.R-project.org/package=randomForest, R package version 4.6-14
Macbeth G, Razumiejczyk E, Ledesma RD (2011) Cliff's Delta Calculator: A Non-parametric Effect Size Program for Two Groups of Observations. Univ Psychol 10:545–555
Madampe K, Hoda R, Grundy J (2020) A Multi-dimensional Study of Requirements Changes in Agile Software Development Projects. arXiv: 2012.03423
Manga I, Blamah NV (2014) A particle Swarm Optimization-based Framework for Agile Software Effort Estimation. Int J Eng Sci 3(6):30–36

Masood Z, Hoda R, Blincoe K (2022) Real World Scrum A Grounded Theory of Variations in Practice. TSE 48(5):1579–1591
Mc Laughlin GH (1969) SMOG Grading-a New Readability Formula. J Read 12(8):639–646
McCallum A, Nigam K (1998) A Comparison of Event Models for Naive Bayes Text Classification. In: Proceedings of the AAAI-98 workshop on learning for text categorization, pp 41–48
McConnell S (2006) Software estimation: demystifying the black art
McHugh ML (2012) Interrater reliability: the kappa statistic. Biochem Med 22(3):276–282
McIntosh S, Kamei Y (2017) Are Fix-Inducing Changes a Moving Target? A Longitudinal Case Study of Just-In-Time Defect Prediction. TSE 44(5):412–428
Menzies T, Dekhtyar A, Distefano J, Greenwald J (2007) Problems with Precision: A Response to "comments on data mining static code attributes to learn defect predictors". TSE 33(9):637–640
Meyer D, Dimitriadou E, Hornik K, Weingessel A, Leisch F (2019) e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. https://CRAN.R-project.org/package=e1071, R package version 1.7-2
Mullen KM, Ardia D, Gil DL, Windover D, Cline J (2011) DEoptim: An R Package for Global Optimization by Differential Evolution. JSS 40(6):1–26
Pasuksmit J, Thongtanunam P, Karunasekera S (2021) Towards Just-Enough Documentation for Agile Effort Estimation: What Information Should Be Documented? In: Proceedings of the ICSME, pp 114–125
Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: Machine Learning in Python. J Mach Learn Res 12:2825–2830
Pinheiro J, Bates D, DebRoy S, Sarkar D, R Core Team (2019) nlme: Linear and nonlinear mixed effects models. https://CRAN.R-project.org/package=nlme, R package version 3.1-141
Popli R, Chauahn N (2015) Managing uncertainty of story-points in agile software. In: Proceedings of the INDIACom, pp 1357–1361
Porru S, Murgia A, Demeyer S, Marchesi M, Tonelli R (2016) Estimating Story Points from Issue Reports. In: Proceedings of the PROMISE, pp 2:1–2:10
Prikladnicki R, Lassenius C, Carver JC (2019) Trends in Agile: From Operational to Strategic Agility. IEEE Softw 36(1):95–97
R Core Team (2013) R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. http://www.R-project.org/
Ramessur MA, Nagowah SD (2021) A predictive model to estimate effort in a sprint using machine learning techniques. IJIT 13(3):1101–1110
Rehurek R, Sojka P et al (2011) Gensim—statistical semantics in python. Retrieved from genism.org
Rennie J, Shih L, Teevan J, Karger D (2003) Tackling the Poor Assumptions of Naive Bayes Text Classifiers. In: Proceedings of the ICML, pp 616–623
Ribeiro MT, Singh S, Guestrin C (2016) Why Should I Trust You?: Explaining the Predictions of Any Classifier. In: Proceedings of the SIGKDD, pp 1135–1144
Rigby PC, Storey MA (2011) Understanding Broadcast Based Peer Review on Open Source Software Projects. In: Proceedings of the ICSE, pp 541–550
Ruangwan S, Thongtanunam P, Ihara A, Matsumoto K (2018) The impact of human factors on the participation decision of reviewers in modern code review. EMSE 24(2):973–1016
Rubin KS (2012) Essential Scrum: A Practical Guide to the Most Popular Agile Process
Sarro F, Petrozziello A (2018) Linear Programming as a Baseline for Software Effort Estimation. TOSEM 27(3):12:1–12:28
Sarro F, Petrozziello A, Harman M (2016) Multi-objective Software Effort Estimation. In: Proceedings of the ICSE, pp 619–630
Sawilowsky SS (2009) New Effect Size Rules of Thumb. JMASM 8(2):597–599
Schwaber K, Sutherland J (2020) The Scrum Guide, The Definitive guide to Scrum: The Rules of the Game
Scott E, Pfahl D (2018) Using Developers' Features to Estimate Story Points. In: Proceedings of the ICSSP, pp 106–110
Sedano T, Ralph P, Péraire C (2019) The Product Backlog. In: Proceedings of the ICSE, pp 200–211
Senter R, E.A.Smith (1967) Automated Readability Index. Tech. rep.
Shah J, Kama N, Bakar NAA (2018) A Novel Effort Estimation Model For Software Requirement Changes During Software Development Phase. IJSEA, 9,(6)
Svensson RB, Gorschek T, Bengtsson P-O, Widerberg J (2019) BAM-Backlog Assessment Method. In: Proceedings of the XP, pp 53–68, pp 53–68
Syed M, Nelson SC (2015) Guidelines for Establishing Reliability When Coding Narrative Data. Emerg Adulthood 3(6):375–387
Tantithamthavorn C, Hassan AE (2018) An Experience Report on Defect Modelling in Practice: Pitfalls and Challenges. In: Proc. of the ICSE-SEIP, pp 286–295

Tantithamthavorn C, Hassan AE, Matsumoto K (2018) The impact of class rebalancing techniques on the performance and interpretation of defect prediction models. TSE 46(11):1200–1219
Tantithamthavorn C, McIntosh S, Hassan AE, Matsumoto K (2017) An Empirical Comparison of Model Validation Techniques for Defect Prediction Models. TSE 43(1):1–18
Tantithamthavorn C, McIntosh S, Hassan AE, Matsumoto K (2019) The Impact of Automated Parameter Optimization on Defect Prediction Models. TSE 45(7):683–711
Tawosi V, Al-Subaihin A, Sarro F (2022) Investigating the Effectiveness of Clustering for Story Point Estimation. In: Proceedings of the SANER), To Appear
Tawosi V, Sarro F, Petrozziello A, Harman M (2021) Multi-Objective Software Effort Estimation: A Replication Study. TSE, To Appear
Therneau T, Atkinson B, Ripley B, Ripley MB (2015) Package 'rpart'. cran.ma.ic.ac.uk/web/packages/rpart/rpart.pdf
Thiele C, Hirschfeld G (2021) cutpointr: Improved Estimation and Validation of Optimal Cutpoints in R. J Stat Softw 98(11):1–27
Thongtanunam P, McIntosh S, Hassan AE, Iida H (2017) Review Participation in Modern Code Review. EMSE 22(2):768–817
Timofeev R (2004) Classification and regression trees (CART) theory and applications. Vol 54, Humboldt University, Berlin,
Tiwari NM, Upadhyaya G, Rajan H (2016) Candoia: A Platform and Ecosystem for Mining Software Repositories Tools. In: ICSE-C, pp 759–761
Tomczak M, Tomczak EWA (2014) The need to report effect size estimates revisited. An overview of some recommended measures of effect size. Trends Sport Sci 21(1):19–25
Torgo L (2010) Data Mining with R, learning with case studies. Chapman and Hall/CRC, London. http://www.liaad.up.pt/~ltorgo/DataMiningWithR
Usman M, Britto R (2016) Effort Estimation in Co-located and Globally Distributed Agile Software Development: A Comparative Study. In: Proceedings of the IWSM-MENSURA, pp 219–224
Usman M, Britto R, Damm L-O, Börstler J (2018) Effort estimation in large-scale software development: An industrial case study. IST 99:21–40
Usman M, Petersen K, Börstler J, Neto PS (2018) Developing and using checklists to improve software effort estimation: A multi-case study. JSS 146:286–309
Vargas EL, Hejderup J, Kechagia M, Bruntink M, Gousios G (2018) Enabling Real-Time Feedback in Software Engineering. In: ICSE-NIER, pp 21–24
Velez DR, White BC, Motsinger AA, Bush WS, Ritchie MD, Williams SM, Moore JH (2007) A balanced accuracy function for epistasis modeling in imbalanced datasets using multifactor dimensionality reduction. Genet Epidemiol 31(4):306–315
von Jouanne-Diedrich H (2017) OneR: One Rule Machine Learning Classification Algorithm with Enhancements. https://CRAN.R-project.org/package=OneR, R package version 2.2
Wilcoxon F (1992) Individual Comparisons by Ranking Methods. In: Breakthroughs in statistics, pp 196–202
Xia T, Shu R, Shen X, Menzies T (2020) Sequential Model Optimization for Software Effort Estimation. TSE. To Appear
Zanetti MS, Scholtes I, Tessone CJ, Schweitzer F (2013) Categorizing Bugs with Social Networks: A Case Study on Four Open Source Software Communities. In: Proceedings of the ICSE, pp 1032–1041
Zimmermann T, Premraj R, Bettenburg N, Just S, Schroter A, Weiss C (2010) What Makes a Good Bug Report? TSE 36(5):618–643



Jirat Pasuksmit is a Ph.D. candidate in Software Engineering at the School of Computing and Information Systems, the University of Melbourne, Australia. He received his B.Sc. in Computer Science from the School of Information and Technology, King Mongkut's University of Technology Thonburi, Thailand. Prior to commencing his Ph.D. study, he was an Agile software developer for four years. His research interests include Agile software development, empirical software engineering, and effort estimation in Agile.

Patanamon Thongtanunam is an ARC DECRA awardee and a lecturer at the School of Computing and Information System, the University of Melbourne, Australia. Prior to that, she was a lecturer at the School of Computer Science, the University of Adelaide, and a research fellow of Japan Society for the Promotion of Science (JSPS). She received Ph.D. in Information Science from Nara Institute of Science and Technology, Japan. Her research interests include empirical software engineering, mining software repositories, software quality, and human aspect. Her research has been published at top-tier software engineering venues like International Conference on Software Engineering (ICSE), Transactions on Software Engineering (TSE), and Empirical Software Engineering (EMSE). More about Patanamon and her work is available online at http://patanamon.com

Shanika Karunasekera received the B.Sc. degree in electronics and telecommunications engineering from the University of Moratuwa, Sri Lanka, in 1990 and the Ph.D. degree in electrical engineering from the University of Cambridge, Cambridge, U.K., in 1995. From 1995 to 2002, she was a Software Engineer and a Distinguished Member of Technical Staff with Lucent Technologies, Bell Labs Innovations, USA. In January 2003, she joined the University of Melbourne, Victoria, Australia, and currently she is a Professor in the School of Computing and Information Systems. Her current research interests include distributed system engineering, distributed data-mining and social media analytics.