Modeling Team Dynamics for the Characterization and Prediction of Delays in User Stories

Kula, Elvan; Deursen, Arie van; Gousios, Georgios

Publication date
2021

Document Version
Accepted author manuscript

Published in
2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)

Citation (APA)
Kula, E., Deursen, A. V., & Gousios, G. (2021). Modeling Team Dynamics for the Characterization and Prediction of Delays in User Stories. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE): Proceedings* (pp. 991-1002). Article 9678939 IEEE.

Modeling Team Dynamics for the Characterization and Prediction of Delays in User Stories

Elvan Kula
Delft University of Technology
Delft, The Netherlands
e.kula@tudelft.nl

Arie van Deursen
Delft University of Technology
Delft, The Netherlands
arie.vandeursen@tudelft.nl

Georgios Gousios
Delft University of Technology
Delft, The Netherlands
g.gousios@tudelft.nl

***Abstract*—In agile software development, proper team structures and effort estimates are crucial to ensure the on-time delivery of software projects. Delivery performance can vary due to the influence of changes in teams, resulting in team dynamics that remain largely unexplored. In this paper, we explore the effects of various aspects of teamwork on delays in software deliveries. We conducted a case study at ING and analyzed historical log data from 765,200 user stories and 571 teams to identify team factors characterizing delayed user stories. Based on these factors, we built models to predict the likelihood and duration of delays in user stories. The evaluation results show that the use of team-related features leads to a significant improvement in the predictions of delay, achieving on average 74%-82% precision, 78%-86% recall and 76%-84% F-measure. Moreover, our results show that team-related features can help improve the prediction of delay likelihood, while delay duration can be explained exclusively using them. Finally, training on recent user stories using a sliding window setting improves the predictive performance; our predictive models perform significantly better for teams that have been stable. Overall, our results indicate that planning in agile development settings can be significantly improved by incorporating team-related information and incremental learning methods into analysis/predictive models.**

I. Introduction

The overall perceived success of a software project depends heavily on the timeliness of its delivery [1]. Reducing delays is therefore a critical goal for software companies. Over the past two decades, software organizations have increasingly embraced agile development methods to manage software projects [2]. Agile gained popularity in the software industry because, in comparison to traditional (waterfall-like) approaches, it uses an iterative approach to software development, aimed at reducing development time, managing changing priorities and inherently reducing risk [3]. However, on-time delivery remains a challenge in agile software development. Prior work [4] has found that around half of the agile projects run into effort overruns of 25% or more.

In agile settings, software is incrementally developed through short iterations to enable a fast response to changing markets and customer demands. Each iteration requires the completion of a number of *user stories*, which are a common way for agile teams to express user requirements. Agile teams are responsible for determining the next iteration's workload together and then breaking these into user stories that can be implemented, tested and shipped in one iteration. Agile teams are characterized by self-organization and intense collaboration [3], [5]. Several studies [1], [6]–[9] have shown the importance of teamwork for the success of agile projects. Various aspects of teamwork, such as team orientation, team coordination and work division, can affect software delivery performance [9], [10]. Moreover, delivery performance can vary due to the influence of changes in teams, resulting in team dynamics that remain largely unexplored. Hence, there is a need to better understand the effects of teamwork and team dynamics on delays, which can benefit the effective application of agile methods in software development.

Today's agile projects require different approaches to planning due to their iterative and team-oriented nature [11]. Central to the planning is the ability to predict, at any phase of the project, if a team can deliver the planned software features on-time. Agile teams would therefore benefit from team-specific, actionable information about the current existence of delay risks at the fine-grained level of user stories, allowing them to take measures to reduce the chance of delays. Recent approaches have leveraged machine learning techniques for evaluating risk factors in software projects (e.g., [12], [13]), estimating effort for issue reports (e.g., [14]–[16]) and predicting delays in bugs or issues (e.g., [17]–[19]). These approaches focus on the technical aspects of software deliveries and do not adequately take into account team-related factors. Studies of software teams [9], [10], [20], [21] have developed theoretical concepts and detailed performance models that articulate relationships between various aspects of teamwork quality and the extent to which a team is able to meet time and cost objectives in software projects. These studies point out that various aspects of teamwork need to be considered when planning software deliveries. Therefore, the predictive power of existing effort prediction models might be enhanced by incorporating such factors.

In this paper, we explore the effects of various aspects of teamwork on delays in software deliveries. Project delays are common in the software industry [4], [22], which makes it important to study this phenomenon in more detail. There is a need to understand and predict, especially during early project phases, which projects will be delayed. This would allow teams to better manage and possibly prevent delays. To do so, we conduct a case study at ING, a large Dutch internationally operating bank with more than 15,000 developers. Teams at ING develop software using an agile development process.

ING offers a great opportunity to study delays in agile projects, as around one quarter of its user stories are delayed. We analyze historical log data from 571 teams and 765,200 user stories at ING to identify team factors characterizing delayed user stories. Based on these factors, we build models that can effectively predict the likelihood and duration of delays in user stories. Our models learn from a team's past delivery performance to predict delay risks in new user stories. To determine whether the use of team features has a positive impact on the predictive performance, we compare the results of models learned using different sets of features: story features, text features and team features. We also evaluate the models with a sliding window setting to explore incremental learning and the impact of team churn on the models' performance. The sliding window works as a forgetting mechanism: the model learns from a team's recent delivery performance in the window and forgets older, irrelevant data to follow team changes over time.

Our results show that the use of team features leads to a significant improvement in the predictions of delay, achieving on average 74%-82% precision, 78%-86% recall, 76%-84% F-measure and 80%-92% AUC. Team features can help improve the prediction of delay likelihood, while delay duration can be predicted exclusively using them. Moreover, developer workload, team experience, team stability and past effort estimates are the most important team features for predicting delay. Finally, training on recent user stories using a sliding window improves the predictive performance; our predictive models perform significantly better for teams that have been stable.

II. Context

In agile software development, a project has a number of iterations (e.g., *sprints* in Scrum [23]). An iteration is usually a short (2–4 weeks) period in which the development team designs, implements, tests and delivers a distinct product increment. Each iteration requires the completion of a number of user stories. Agile teams work with a product backlog to keep track of the status and priority of user stories [24]. Figure 1 shows an example of a user story from the backlog management tool used by teams at ING. A user story has a title, textual description and a few standard fields to record its priority, type, status and dependencies on other stories.

Planning is done before an iteration starts and focuses on selecting and estimating the user stories to be delivered in that iteration. To plan the iteration, the team discusses each story and breaks it down into tasks to facilitate estimation [26]. Multiple developers can work on various sub-tasks of a story but only one developer is assigned to the story and responsible for its implementation. Agile teams heavily rely on experts' subjective assessment to estimate the effort of completing a user story [27]. *Story points* are a commonly used unit of measure that reflect the relative amount of effort, complexity and risks involved in implementing the user story [11]. Agile teams usually estimate story points together in a dedicated planning session (e.g., using Planning Poker [28]).

A. Usage Scenarios

At the end of an iteration, a number of user stories are completed and there may also be a number of incomplete/unresolved user stories delayed to future iterations. Our prediction models enable teams to identify these user stories before the start of an iteration. There are two scenarios in which predictions are being made: *before* and *after* an effort estimate has been made for a user story. The availability of an estimate might affect the accuracy and usefulness of our predictions. It is likely that in the latter scenario, our predictions get more accurate (since we have information about the estimated size of a user story) but the less useful it is (since the team has already spent a considerable amount of time on estimating the story).

In both scenarios, our predictive models can be used as a decision support system to generate proactive feedback and make informed decisions on the planning and feasibility of a user story. Foreseeing delay risks allows teams to identify problematic user stories and take corrective actions, such as story splicing (i.e., splitting large stories into smaller ones) or resolving inter-story dependencies. Our models learn from the past delivery performance of the specific team which they are deployed to assist. Hence, the predictions our models make are team-specific. This helps teams improve their schedule estimates and gain an increased awareness of their own behavior patterns.

B. Teams and User Stories at ING

In recent years, ING has reinvented its organisational structure, moving from traditional functional departments to a completely agile organisational structure based on Spotify's '*Squads, Tribes and Chapters*' model [29]. The main purpose of this model is to be able to control agile with hundreds of development teams. All development teams at ING use Scrum as agile methodology. They work with sprints of one to four weeks. The teams consist of 5 to 9 members, including a Scrum master and product owner. The product owner is responsible for prioritizing the product backlog. In consultation with the product owner, the teams divide up the

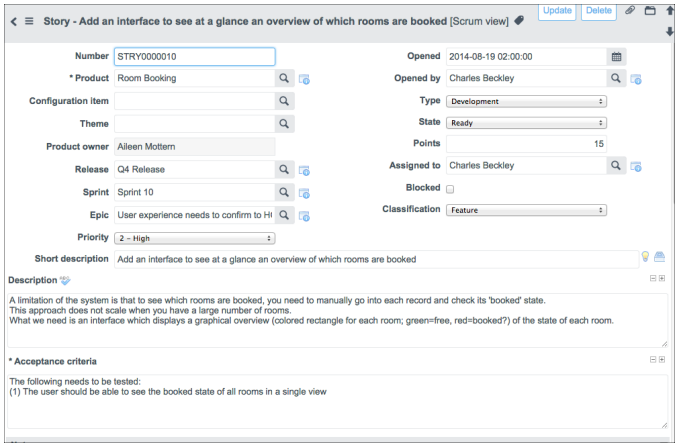


Fig. 1: An example of a user story [25]

work to be done into user stories. Story points are assigned using planning poker [28] in a structured group meeting called *Sprint Planning* before the start of the sprint.

III. Study Design

In this paper, we propose a team-driven approach to determine early on the impact and probability of a delay risk occurring in a user story. To do so, we extract 24 risk factors representing technical and team-related aspects of a user story. Based on these factors, we build models that can effectively predict the likelihood and duration of delays in user stories. As discussed in Section II, predictions are made for two usage scenarios: before and after an effort estimate has been made for the user story. For convenience, in the remainder of this paper, we denote the scenarios as SC1 and SC2, respectively.

Throughout our study, the following research questions guide our work:

- **RQ1. Benefits of team features:** *Does the use of team features have a positive impact on our predictive performance? (RQ1.1), How effective is our approach when team features are used exclusively? (RQ1.2)* To answer these questions, we compare the performance of models learned using combinations of different sets of features: story features, text features and team features.
- **RQ2. Feature importance:** *Which team features are most important for predicting delays in user stories?* For this question, we train models using the extracted 24 features and determine the relative importance of team features in terms of predictive power.
- **RQ3. Benefits of sliding window:** *Does the use of a sliding window provide more accurate and robust estimates?* As teams change over time and these changes might affect their delivery performance, we want to analyze whether it is beneficial for our predictive model to learn from recent user stories and forget older data. To do so, we compare the performance of models learned using all features in a sliding window setting versus expanding window setting.
- **RQ4. Factor of change:** *How does team churn affect story delays and our predictive performance?* Changes in team composition (due to either a member leaving or joining the team) can cause teams to become less predictable at delivering software. We employ the sliding window setting and determine for each user story the number of consecutive windows a team has been stable for. We perform a statistical analysis to assess the impact of the number of consecutive stable windows on story delays and our models' performance.

We can split our approach into four main steps:

1) *Data collection and pre-processing:* We collect and preprocess backlog management data (past user stories) from 571 development teams at ING.
2) *Risk factor extraction and analysis:* We extract 24 risk factors representing technical and team-related aspects of user stories, and then perform correlation analysis to determine whether the factors affect delays in user stories.
3) *Text feature extraction:* We use RoBERTa [30], a state-of-the-art language representation model, to produce vector representations (i.e., embeddings) for the textual descriptions of user stories. To adapt the model to our prediction task, we update it with additional training on our corpus of unlabeled user stories.
4) *Model building:* We use the selected risk factors and text embeddings to build models that predict delays in user stories.
5) *Model evaluation:* We evaluate our models using various sets of features in different experimental settings to answer the research questions.

A. Data Collection and Pre-Processing

We extracted log data from *ServiceNow*, a backlog management tool used by a majority of teams at ING [25]. The dataset consists of user stories delivered by 571 teams at ING between January 01, 2016 and January 01, 2021. The user stories have significant variety in terms of the products developed, the size and application domain (banking applications, cloud software, software tools). The dataset contains the following fields for user stories: *Identification Number*, *Creation Date*, *Sprint Identification Number*, *Planned Start Date*, *Actual Start Date*, *Planned Delivery Date*, *Actual Delivery Date*, *Story Points* and the textual *Title* and *Description* fields. The *Planned Start Date* coincides with the start date of the sprint that the story was originally assigned to. We acknowledge that the planned start date of a user story might change before the sprint is started. Therefore, we consider only the planned start date as scheduled on the day that the development phase of a sprint is started. For each user story, the dataset contains the entire history of changes. This enabled us to track the number of sprints a user story was delayed for. We acknowledge that a team might decide to temporarily move a story back to the product backlog after a delay. Therefore, we calculate the delay duration based on the number of sprints a story has actually been part of.

To eliminate noise and missing values, we removed user stories with a status other than 'Completed'. We also filtered out user stories with empty *Planned Delivery Date*, *Actual Delivery Date*, *Story Points* and *Description* fields. Moreover, we deleted user stories that have not been assigned to a developer. We also removed stories that were added to a sprint during the development phase, because they are likely to be unstable and not accurately represent delay. We found a few user stories that had been delayed for an unusually long period of time (e.g., in some cases over 10 sprints). We removed such outliers that exceed two standard deviations from the mean delay duration of all user stories. The original dataset contained 889,014 user stories. After removing outliers and pre-processing the data, the final dataset decreased to 765,200 user stories from 571 teams. This dataset consists of 183,342 (24%) delayed and 581,858 (76%) non-delayed user stories.

TABLE I: The 24 extracted risk factors representing the characteristics of a user story and development team. Correlation coefficients are based on Spearman's Correlation [32]: they measure the strength of the relationship between factors and the risk classes. Statistical significance with Holm correction [33] is indicated with * (p-value < 0.01) and ** (p-value < 0.001).

Category	Factor name	Description	Type	Correlation coefficient	
				Spearman's ρ	Interpretation
Story factors	dev-type	The development type (1. new feature, 2. bug fix or 3. improvement) of a story	Categorical	−0.19*	Weak
	priority	Does a story have a major priority to the customer?	Binary	−0.31**	Weak
	security	Whether a story is associated with a security-critical system	Binary	0.44**	Moderate
	out-degree	Number of outgoing dependencies of a story on other stories	Continuous	0.41**	Moderate
	sprint-duration	Planned duration of the sprint that a story was originally assigned to	Continuous	−0.26**	Weak
	planned-stories	Total number of user stories in the sprint that a story was originally assigned to	Continuous	0.22**	Weak
	planned-points	Total number of story points in the sprint that a story was originally assigned to	Continuous	0.13**	Weak
	initial-points	Number of story points initially estimated for a user story	Continuous	0.51**	Moderate
Team factors	team-size	Number of team members	Continuous	0.08*	Weak
	avg-story-size	Average number of story points that the team assigned to past stories	Continuous	0.46*	Moderate
	team-existence	Number of years the team has existed for	Continuous	−0.35**	Weak
	team-stability	Ratio of team members that did not change in the last six months	Continuous	−0.40**	Moderate
	po-stability	Did the product owner of the team stay the same in the last six months?	Binary	−0.29**	Weak
	team-capacity-stories	Total number of user stories that have been completed by the team so far	Continuous	−0.28**	Weak
	team-capacity-points	Total number of story points that have been completed by the team so far	Continuous	−0.26**	Weak
	global-distance	The Global Distance Metric [31] measured across teams members	Continuous	0.17*	Weak
	dev-seniority	The seniority rank of a developer at ING	Categorical	0.24**	Weak
	dev-age-team	Number of years spent by a developer in the current team	Continuous	−0.28**	Weak
	dev-age-project	Number of years spent by a developer in the current project	Continuous	−0.12**	Weak
	dev-age-abc	Number of years spent by a developer at ING	Continuous	−0.43**	Moderate
	dev-workload-stories	Number of user stories assigned to a developer in the current sprint	Continuous	0.53**	Moderate
	dev-workload-points	Number of story points assigned to a developer in the current sprint	Continuous	0.49**	Moderate
	dev-capacity-stories	Total number of user stories that have been completed by a developer so far	Continuous	−0.45**	Moderate
	dev-capacity-points	Total number of story points that have been completed by a developer so far	Continuous	−0.41**	Moderate

Risk classes. The delayed stories in our dataset consist of 76,398 (42%) stories that were delayed for a single sprint, 61,052 (33%) stories that were delayed for two sprints, 34,821 (19%) stories that were delayed for three sprints and 11,071 (6%) stories that were delayed for more than three sprints. For our predictions, we choose to use four risk classes that reflect the degree of delay: *non-delayed*, *minor delay* (delay of one sprint), *medium delay* (delay of two sprints) and *major delay* (delay of three sprints or more). Since a small fraction of the user stories in our data were delayed for more than three sprints, we decided to merge this group with the user stories that were delayed for three sprints.

B. Risk Factor Extraction and Analysis

We extracted 24 risk factors from the collected log data to explore which factors characterize delayed user stories. Table I provides an overview and correlation analysis of these factors. The factors are divided in two groups: *story factors* and *team factors*. The story factors represent the inherent characteristics of user stories, such as its size, type and priority. The team-related factors represent characteristics of individual team members and the group as a whole. We now explain the risk factors in detail. The factor names are underlined.

Story factors. Several story factors are extracted directly from the story's primitive attributes, which include dev-type and priority. Each story will be assigned a type and priority which indicate the nature and urgency of the task associated with implementing the story. Both factors have been shown to affect the delivery of a story in related work [34]. We extract security to determine whether a user story needs to go through a mandatory, resource-intensive security testing procedure at ING that might lead to delay. We extract the outgoing degree (out-degree) of dependencies of a story; this has been shown to predict delay in related work [17]. The remaining story factors are used to extract the size of a story and the sprint.

Team factors. Previous work (e.g. [35], [36]) has found that member turnover can lead to tacit knowledge loss, and thus may negatively affect team productivity. Therefore, we compute the stability of a team (team-stability) and that of its product owner (po-stability). We also measure team-existence to quantify the familiarity and maturity of a team; both have been shown to lead to better team interactions and project performance in related work (e.g., [37], [38]). Previous studies (e.g., [39], [40]) have shown that the interactions among team members are less effective in distributed teams. To quantify the distance between team members, we calculate global-distance based on the Global Distance Metric proposed in related work [31]. We calculate the metric for pair-wise combinations of team members and take the maximum value.

Developers' capabilities and experience can influence their contributions to projects (e.g., [36], [41], [42]). Thus, we compute dev-capacity-stories and dev-capacity-points to quantify the software delivery experience of the developer that a user story is assigned to. Similarly, we use team-capacity-stories and team-capacity-points to quantify the team's overall experience with software deliveries. Moreover, we extract the developers' seniority; the intuition here is that senior developers might more often be assigned to complex user stories that have a higher delay risk. ING employs the five-stage Dreyfus Model [43] to assess the expertise of developers based on their experience in the software industry. We also extract dev-age-team, dev-age-project and dev-age-abc to measure how long team members have been working in their specific teams, projects and at ING.

Related work has identified an inappropriate division

of work as an important barrier to achieving team effectiveness [9]. Hence, we calculate dev-workload-stories and dev-workload-points. Finally, we extract team-size and avg-story-size as larger projects are associated with greater risk in literature (e.g., [37], [44], [45]).

C. Text Feature Extraction

The title and description of a user story can provide good features since they explain the nature and complexity of a story. To extract features from text, we combined the title and description of a user story into a single text document where the title is followed by the description. Our approach computes vector representations for these documents that are then used as features to predict delays in user stories. We tried different methods for text feature extraction: the traditional Bag-of-Words (with *TF-IDF* [46] and *Okapi BM25* [47]), the neural network-based *Doc2Vec* [48] and the state-of-the-art transformer-based language model *RoBERTa* [30]. In case of TF-IDF and BM25, we pre-processed the texts by lower casing the words and removing punctuation and stop words. We compared and evaluated the methods on our prediction task and corpus of user stories. We found that RoBERTa outperforms the other methods on average by 11%-27% in precision, 5%-38% recall and 9%-33% F-measure. Therefore, we decided to use RoBERTa as part of our experimental setup for answering the research questions.

RoBERTa is an optimization based on Google's BERT [49], which is able to learn bidirectional word embeddings from texts. To adapt the model to the domain-specific vocabulary of user stories, we updated it with additional training on our corpus of unlabeled user stories. In this updating procedure, we implemented the same masked language modeling strategy as in the pre-training procedure of the original model, with a set of newly designated hyperparameters (*training steps: 60K, batch size: 64, optimizer: Adam, learning rate 3×10-5*).

D. Model Building

Our objective is to predict the probability of a delay occurring (i.e., delay likelihood) and a probability distribution over the aforementioned risk classes (i.e., delay duration in terms of the number of sprints overrun). Therefore, our predictive models should be able to provide probability estimates. We employ *binary classification* for predicting the likelihood of delay. We reduce the aforementioned risk classes into two binary classes: *delayed* and *non-delayed*. The delayed class covers the user stories that belong to the aforementioned *minor delay*, *medium delay* and *major delay* classes. For predicting the delay duration of delayed stories, we employ *multi-class classification* for the *minor delay*, *medium delay* and *major delay* classes.

We compared and evaluated four different classifiers that are able to provide class probabilities and that have been shown to be effective classifiers in risk prediction: Random Forests [50], AdaBoost [51], Multi-layer Perceptron [52] and Naive Bayes [53]. A comparison on both prediction tasks showed that Random Forests outperforms the other classifiers on average by 2%-23% in precision, 9%-48% in recall and 4%-33% in terms of F-measure. Therefore, we chose to employ Random Forests (RF) [50] as part of our experimental setup.

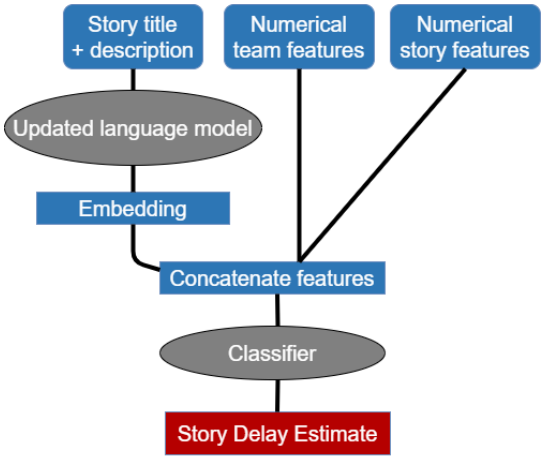


Fig. 2: Our pipeline for predicting delays in user stories

Figure 2 shows the design of our pipeline of predicting delays in user stories: (i) extract numerical story features, (ii) extract numerical team features, (iii) produce document representations of user stories using RoBERTa, (iv) concatenate features and (iv) classification using Random Forests.

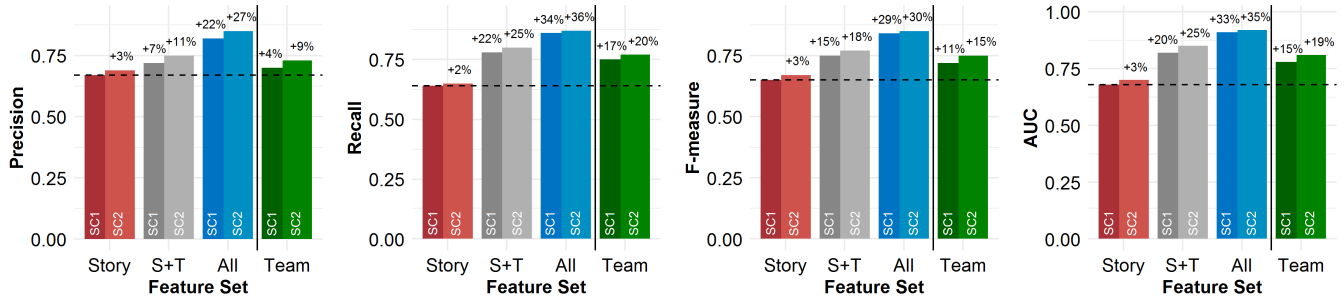
E. Model Evaluation

Evaluation setup. We performed experiments on the 765,200 user stories in our dataset. We built team-specific predictive models, meaning that our models are trained and tested on a dataset containing the past user stories from one specific team. Hence, we built two models for each team in the dataset: one for predicting delay likelihood and one for predicting delay duration.

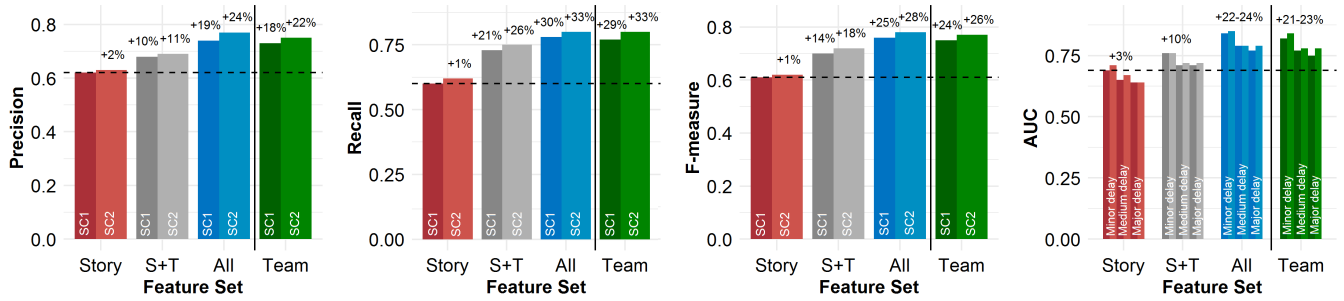
To mimic a real prediction scenario where the delay of a given user story is estimated based on knowledge from previous stories, we sorted the stories based on their start date. Then, for training and evaluation in RQ1 and RQ2, we used time-based 10-fold cross-validation. Cross-validation is a well-known technique to prevent the classifier from overfitting. The time-based variant of cross-validation ensures that in the kth split, the stories in the first k folds (training set) are created before the stories in the (k+1)th fold (test set). Unlike standard cross-validation, successive training sets are supersets of previous ones (also known as an expanding window).

For RQ1, we evaluated the performance of the models learned using different sets of features. We ran all experiments for the two types of usage scenarios: SC1 (excluding the initial-points feature) and SC2 (including the initial-points feature). For RQ2—RQ4, we ran the experiments using all features (including initial-points).

Our predictive models are able to estimate class probabilities for the delay likelihood and delay duration of user stories. During the testing phase of our models, we chose the class with the highest probability as the predicted class.



(a) Evaluation results obtained for predicting delay likelihood: the *story features* baseline achieved 0.67/0.69 precision, 0.64/0.65 recall, 0.65/0.67 F1 and 0.68/0.70 AUC for SC1/SC2.



(b) Evaluation results obtained for predicting delay duration: the *story features* baseline achieved 0.62/0.63 precision, 0.60/0.62 recall, 0.61/0.62 F1, 0.69/0.71 $\text{AUC}_{\text{minor delay}}$, 0.65/0.67 $\text{AUC}_{\text{medium delay}}$ and 0.64/0.64 $\text{AUC}_{\text{major delay}}$ for SC1/SC2.

Fig. 3: Evaluation results for predicting the likelihood and duration of delay in usage scenario SC1 (before effort estimation) and SC2 (after effort estimation) using *story features*, a combination of *story and text features* (S+T) and *all features* (story, text and team features) (RQ1.1). The results are averaged across the teams in the dataset. The predictions of the *story features* model are used as a baseline and visualized as a dashed line. Results are also given for when team features are used exclusively (RQ1.2).

Sliding window setting. For RQ3, we evaluated our predictive models using two different experimental settings: the expanding window and the sliding window. In both settings, the user stories are first sorted based on their start date and then divided into multiple time-based windows. For each window k_i in the expanding window setting, we use the stories from the previous windows $k_0 ... k_{i-1}$ to train a model. In the sliding window setting, however, we train the model only on the last window k_{i-1}. The sliding window allows us to train the model on a team's recent delivery performance, while the expanding window uses all observations available.

To analyze the impact of team churn on delays and our models' performance (RQ4), we employed the sliding window setting and determined for each window whether a team had been stable or not. We marked a team as stable during a window if no team members left or joined the team during that window (i.e., if the team composition did not change during that window). Then, for each story, we determined the number of consecutive windows a team had been stable for. If the team composition had changed in the previous window, then this number was considered to be zero. Finally, we performed a statistical comparison of delays and our models' evaluation results between stable and unstable teams.

Performance measures. We computed the widely used precision, recall and F1-score to evaluate the performance of our predictive models. To account for class imbalance, we calculated the weighted averages of these measures (i.e, the score of each class is weighted by the number of samples from that class). We also used Area Under the Curve (AUC) of receiver operator characteristics (ROC) [54] in classifying the outcome of a user story.

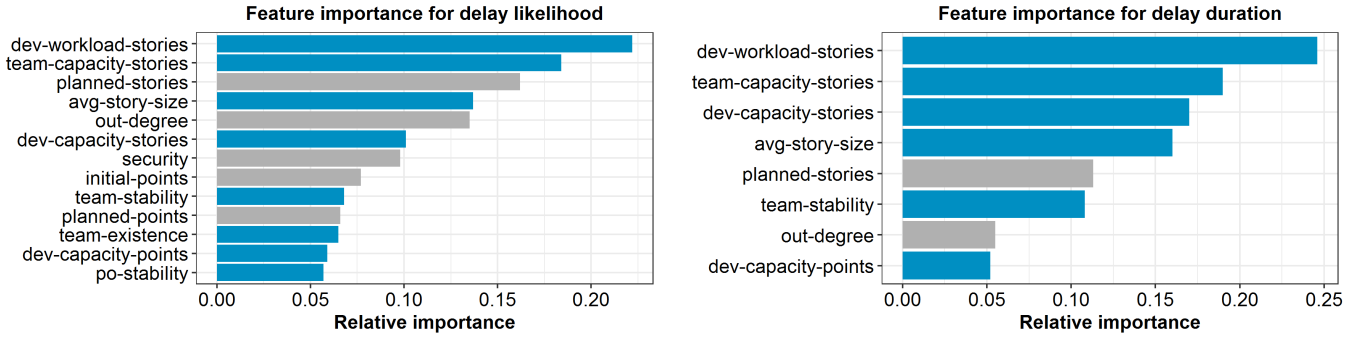
To compare the performance of predictive models, we tested the statistical significance of their evaluation results using the Wilcoxon Signed Rank Test [55]. This is a non-parametric test that makes no assumptions about underlying data distributions. We employed the non-parametric effect size measure, the Vargha and Delaney's $\hat{A}_{12}$ statistic [55]. This measure is commonly used for evaluation in effort estimation [56].

IV. Results

In this section, we report the results in answering research questions RQs 1-4.

RQ1: Benefits of Team Features

For this research question, we compared the performance of models learned using story features, a combination of story and text features, and all features (story, text and team features). We used the Wilcoxon test and $\hat{A}_{12}$ effect size to investigate whether the improvements achieved by the addition of team features are statistically significant.



(a) (b)

Fig. 4: Feature importance for predicting delays in user stories: team features are highlighted in blue, story features in gray.

RQ1.1: Does the use of team features have a positive impact on our predictive performance? Figure 3a presents the evaluation results for predicting delay likelihood in usage scenarios SC1 and SC2 (described in Section II-A). In both scenarios, the models learned using all features outperform the models learned using a subset of features in terms of precision, recall, F1 and AUC. On average, the all-features models improve the *story* and *story+text* models by 19% (precision), 23% (recall), 21% (F1-score) and 22% (AUC). Statistical tests show that the improvements achieved by the all-features models are significant ($p < 0.001$) and the effect sizes are large (ranging between 0.71 and 0.87). *This demonstrates that the addition of team features leads to a significant improvement in the predictions of delay likelihood.*

Figure 3b presents the evaluation results for predicting delay duration. Similarly, in both scenarios, the models learned using all features outperform the models learned using a subset of features in terms of precision, recall, F1 and AUC. On average, the all-features models improve the *story* and *story+text* models by 16% (precision), 20% (recall), 18% (F1-score) and 17% (AUC). These improvements are significant ($p < 0.001$) and the effect sizes are at least medium (ranging between 0.65 and 0.78). *This indicates that the addition of team features leads to a significant improvement in the predictions of delay duration.*

RQ1.2: How effective is our approach when team features are used exclusively? As shown in Figure 3a, the models learned using team features only for predicting delay likelihood perform better than the *story* models and slightly worse than the *story+text* models. The improvements achieved by the *all-features* models over the *team* models are significant and the effect sizes are at least medium (ranging between 0.63 and 0.76). *This indicates that the three feature sets have statistically significant contributions to the predictions of delay likelihood.*

Figure 3b shows that the models learned using team features only for predicting delay duration achieve similar results as the *all-features* models. The statistical tests show that the differences between both models in terms of precision, recall, F1 and AUC are significant but the effect sizes are negligible. *This indicates that we can effectively predict delay duration using team features exclusively.*

RQ2: Feature Importance

Using the feature importance evaluation built in Random Forests [57], we obtained the top most important features and their normalized weights from models learned using story and

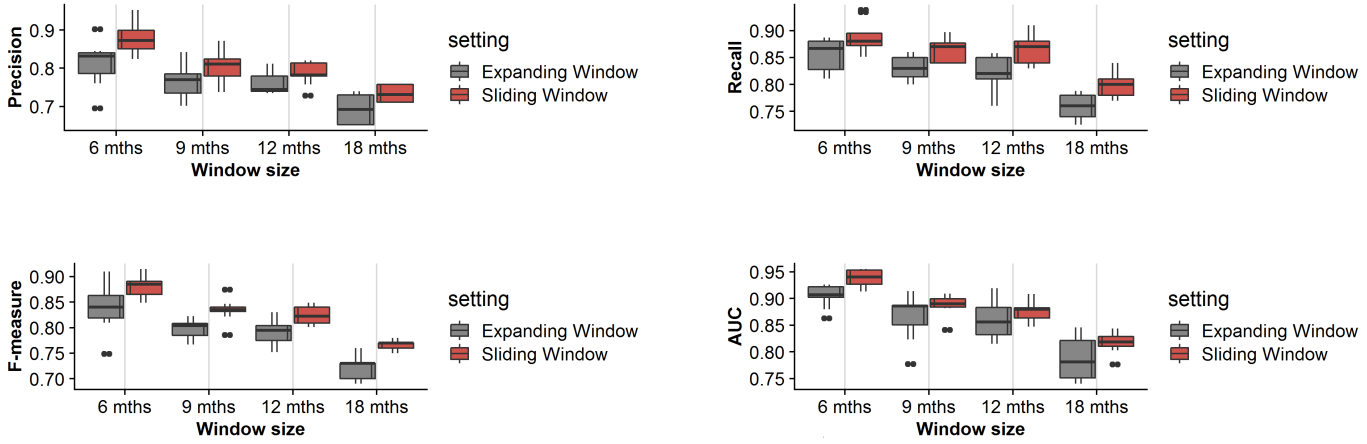
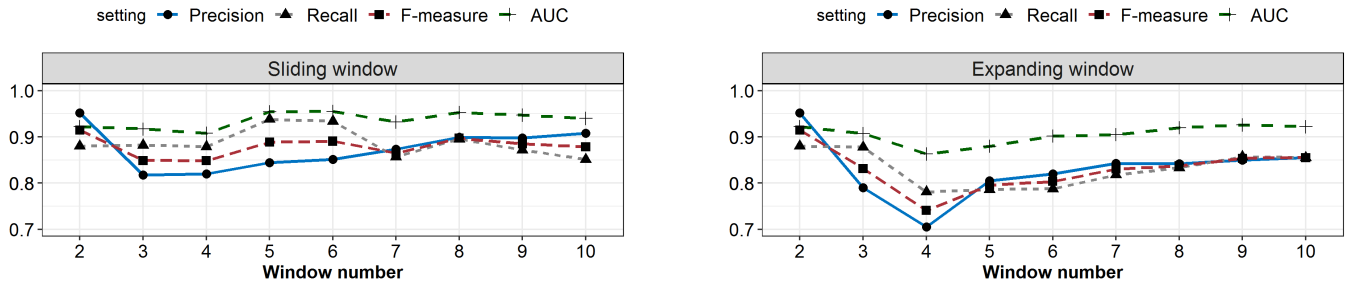


Fig. 5: Evaluation results for predicting delay likelihood across different window sizes in a sliding versus expanding window setting.



(a) Sliding window

(b) Expanding window

Fig. 6: Evaluation results obtained over time in the sliding window and expanding window settings (using a 6-month window)

team features (including initial-points). The text features were not included as it is not possible to reduce the vectors produced by RoBERTa to one single feature. The models learned using story and team features achieve on average 0.74/0.76 precision, 0.79/0.80 recall, 0.77/0.78 F1 and 0.81/0.81 AUC for predicting delay likelihood/duration. Figures 4a and 4b provide a ranking of the features by order of importance for predicting delay likelihood and delay duration. We averaged the importance values of the features across the teams in the dataset to produce an overall ranking. Features that have an importance value lower than 0.05 are not shown.

Figure 4a shows that 13 features from Table I contribute significantly to the predictions of delay likelihood. Dev-workload-stories, team-capacity-stories, planned-stories, avg-story-size and out-degree are the top-5 most important features. Their importance values range from 14% to 22%. Figure 4b shows that a partially overlapping set of 8 features is effective in predicting delay duration. Even though the top most important features in Figure 4b are similar to those for delay likelihood, there are a few ranking differences. Overall, the team features have greater explanatory power for delay duration than for delay likelihood. This corresponds to our results for RQ1. Dev-capacity-stories and team-stability play a significantly larger role in the predictions of delay duration.

RQ3: Benefits of Sliding Window

Figure 5 presents the evaluation results obtained for predicting the likelihood of delay using an expanding versus sliding window. The results are averaged across windows and the teams in the dataset. As shown in Figure 5, the sliding window consistently outperforms the expanding window in terms of precision, recall, F1 and AUC across all window sizes. The Wilcoxon test shows that the improvements are significant ($p < 0.001$), and the effect sizes are between 0.55 and 0.68 (small to medium). Comparing the results across window sizes, we observe that both the expanding window and the sliding window achieve the best performance for a window of six months.

Figures 6a and 6b visualize the evaluation results obtained over time in the sliding and expanding window settings using a 6-month window. The first window is not included as it is used for training only. Figure 6a shows lower variance over time in the prediction results for the sliding window. Both window settings start off with the same precision, recall, F1 and AUC scores for the second window and then their performance declines during the third and fourth windows. Further analysis of the data shows that a majority of teams have greater variance in their story delays during the initial windows. This might explain why the performance of both approaches decline at the start. We observe that the performance of the expanding window drops drastically during the initial windows, while the performance of the sliding window remains more stable. This suggests that the sliding window is better able to adapt to changes in teams' delivery performance.

RQ4: Factor of Change

Figure 7 presents a percentage distribution of different levels of team stability based on the percentage of stories that were delayed. We observe that user stories that are delivered by stable teams are less likely to be delayed. 29% of the stories that have been delivered after a team change (i.e., zero stable windows) are delayed, of which 10% are hindered by a major delay. The percentages of delayed user stories decrease for teams that have been stable for a longer period of time. 21% of the stories that have been delivered after one stable window are delayed, of which 7% has a major delay. Only 15%-17% of the stories that have been delivered after more than one stable window are delayed.

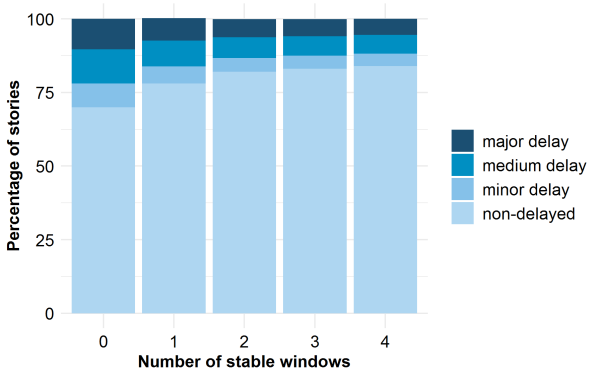


Fig. 7: Delay percentage distribution across different stability levels

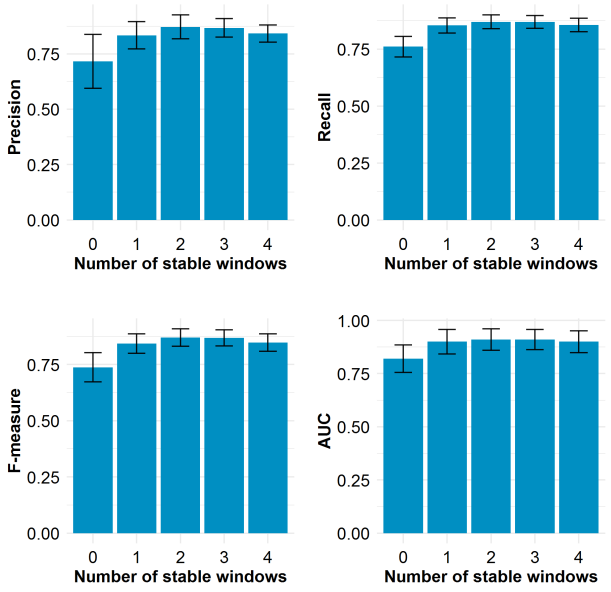


Fig. 8: Evaluation results obtained for different levels of stability

Figure 8 presents the evaluation results obtained for predicting the likelihood of delay for stories delivered by teams of varying stability. We observe that our predictive models achieve better precision, recall, F1 and AUC scores for teams that have been stable. On average, the models achieve 19% higher precision, 13% higher recall, 16% higher F1 and 10% higher AUC for stories that are delivered after at least one stable window. Statistical tests show that these improvements are significant ($p < 0.001$) and the effect sizes are between 0.63 and 0.71 (small to medium). Figure 8 also shows a greater variance in the results for stories delivered after a team change.

V. Discussion

A. Recommendations for Practitioners

Our study provides practitioners with an extensive list of risk factors. By collecting and analyzing these factors, software companies can identify delay risks and derive useful models to predict delays in deliveries. Our models are effective in predicting the likelihood and duration of delays in user stories, achieving on average 74%-82% precision, 78%-86% recall, 76%-84% F-measure and 80%-92% AUC. Our models enable development teams to foresee, either before or after effort estimation, if a user story is at risk of being moved to a future iteration. This allows teams to identify problematic user stories and take corrective actions to reduce the chance of delays.

The evaluation results of our predictive models show that the use of team features leads to a significant improvement in the predictions of delay. Moreover, our results show that team features can help improve the prediction of delay likelihood, while delay duration can be explained exclusively using them. For delay likelihood, the feature sets are complementary to each other. This means that the probability of a delay occurring depends on a combination of technical and team-related aspects, while the impact of the occurred delay (i.e., how fast it is resolved) mainly depends on the characteristics of the team. We therefore recommend organizations to encourage and facilitate teams to improve their work allocation, effort estimates, knowledge sharing and accountability in order to reduce the impact of delays. Companies must also have a stable ecosystem in place to ensure that teams are able to operate effectively.

B. Implications for Researchers

Feedback mechanisms on team behaviors. Our predictive models can be used by teams as awareness or feedback mechanisms on their behavior patterns during planning. The support provided by our models can help teams to increase the awareness of their own behavioral habits (e.g., to reduce the bias towards over-optimistic estimates). This is likely to foster productive behavior change and improve schedule estimates. To better realize the benefits of such mechanisms, there is a need for better tool support that can support agile teams in tracking their team behavior and improving the management of agile projects. Current agile project management tools lack advanced analytical methods that are capable of deriving actionable insights from project data for planning. An extension of existing tools with actionable information about team dynamics and the current existence of risks in a sprint would be beneficial. Initial work in this direction has been carried out by Kortum et al. [58].

Team dynamics monitoring. One of the key novelties in our approach is deriving new team features for a user story by aggregating the features either at team- or individual-level. We derived the features by using a range of statistics over the past stories delivered by a specific team or developer. Our experimental results demonstrate the effectiveness of this approach. Previous research [59], [60] has shown that team-related information is often difficult to capture or not available due to lacking information sources. As a consequence, these factors are often not monitored. Our results indicate the significant benefits of incorporating such data into analysis/predictive models for effort estimation and planning in agile projects. Further research on team monitoring approaches is needed to address this gap and to gain a better understanding of what information and metrics can be collected across software organizations.

Impact of social-driven factors. The set of team-related factors identified in this paper are by no means comprehensive to encompass all aspects of teamwork. We were limited to the repository data available at ING. It is an interesting opportunity for future work to analyze the effects of social-driven factors related to the collaborative nature of software development work. Previous studies [8], [40], [61] have reported on the impact of social-driven factors, such as trust, team leadership, team cohesion and communication. These factors would be a good starting point for future work.

Applicability of effort prediction models. Our evaluation results show that our predictive models perform significantly

better for teams that have been stable for a longer period of time. In our analyses, we did not make a distinction between someone leaving or joining the team, nor did we take into account the number of people leaving or joining. It is an interesting opportunity for future work to analyze the effects of different types of team changes on the models' performance. This could also include external changes, such as organizational restructuring or changes in senior management. Such changes have been identified as risk factors in literature [62]–[64]. Future research should also explore the impact of other team characteristics (e.g., team experience and seniority) on the performance of effort prediction models.

VI. Threats to Validity

Construct validity. We consider data variables as constructs to meaningfully measure delay and risk factors. This introduces possible threats to construct validity [65]. We mitigated these threats by collecting real-world data from user stories and teams, and all the relevant historical information available. The ground-truth (i.e., the delay in terms of the number of sprints overrun) is based on the number of sprints a story has been part of. However, it might happen that teams close their stories too early or too late. We cannot account for the impact of poor record keeping on our results. Even though all teams at ING are encouraged to deliver on-time, there is a possibility that some teams treat their delivery deadlines less seriously than others. These teams might add stories to sprints without the commitment to deliver on-time.

Internal validity. Our dataset has the class imbalance problem. This has implications to a classifier's ability to learn to identify delayed stories. To overcome this issue, we used the weighted variants of performance measures and employed AUC which is insensitive to class imbalance. We applied statistical tests to verify our assumptions [55], and followed best practices in evaluating and comparing predictive models for effort estimation [66]–[68]. However, we acknowledge that more advanced techniques could also be used, such as statistical over-sampling [69]. Another threat to our study is that the patterns in the training data may not reflect the situation in the test data. To mitigate this threat we used time-based cross-validation to mimic a real prediction scenario.

External validity. As with any single-case empirical study, external threats are concerned with our ability to generalize our results. We have considered 765,200 user stories from 571 teams, which differ significantly in size, composition, products developed and application domain. Although we control for variations using a large number of projects and teams, we acknowledge that our data may not be representative of software projects in other organizations and open source settings. Agile teams in other settings might have a different team structure and may work at a different pace or create stories differently. Further research is required to confirm our findings in different development settings.

VII. Related Work

Teamwork in agile development. Previous research has analyzed the nature of agile teams in software development: their characteristics, how they collaborate, and the challenges they face in geographically and culturally diverse environments [70], [71]. A survey of success factors of agile projects identified team capability as a critical factor [1]. Other studies [9], [21] have used performance models to investigate the impact of teamwork quality on project success. Moe et al. [9] showed that problems with team orientation, coordination, work division and conflict between team and individual autonomy are important barriers for achieving team effectiveness. Melo et al. [36] found that team structure, work allocation and member turnover are the most influential factors in achieving productivity in agile projects. Our study complements prior work by exploring the effects of various aspects of teamwork in the context of predicting software delays.

Effort estimation and planning. A great body of research has been published on the study of effort estimation methods [72]. Estimation methods that rely on expert's subjective judgement are most commonly used in agile projects [4]. Project factors and personnel factors are the top mentioned effort drivers in agile projects [4], [27].

Recent efforts have leveraged machine learning techniques to support effort estimation and planning in software projects. They have achieved promising results in estimating effort involved in resolving issues [14]–[16], predicting the elapsed time for bug-fixing or resolving an issue (e.g., [19], [73]–[75]). Previous research [17], [18], [34] has also been done in predicting the delay risk of resolving an issue in traditional development. Some studies have been dedicated to effort estimation for agile development at the level of issues [14]–[16] and iterations [76], [77]. Our work specifically focuses on predicting delays in user stories, and complements prior work by introducing team features and incremental learning to follow team changes over time.

VIII. Conclusion

Modern agile development settings require different approaches to planning due to their iterative and team-oriented nature. In this paper, we explored the effects of various aspects of teamwork on delays in software deliveries. We extracted a set of technical and team-related risk factors that characterize delayed user stories. Based on these factors, we built models that can effectively predict the likelihood and duration of delays in user stories. The evaluation results demonstrate that:

1) The use of team features leads to a significant improvement in the predictions of delay likelihood, while delay duration can be predicted exclusively using them.
2) Developer workload, team experience, team stability and past effort estimates are the most important predictors for delays in user stories.
3) Training on recent user stories leads to more accurate and robust predictions of delay.
4) Our predictive models perform significantly better and more consistently for teams that have been stable.

Overall, our results indicate that planning in agile software development can be significantly improved by incorporating team-related information and incremental learning methods into analysis/predictive models. We identified several promising research directions related to team dynamics monitoring, designing feedback and awareness mechanisms on team behaviors, and the applicability of effort prediction models in agile projects. Progress in these areas is crucial in order to better realize the benefits of agile development methods.

REFERENCES

[1] T. Chow and D.-B. Cao, "A survey study of critical success factors in agile software projects," *Journal of systems and software*, vol. 81, no. 6, pp. 961–971, 2008.

[2] VersionOne, "14th state of agile survey," https://stateofagile.com/#ufh-i-615706098-14th-annual-state-of-agile-report/7027494, accessed: 2021-03-28.

[3] A. Cockburn and J. Highsmith, "Agile software development, the people factor," *Computer*, vol. 34, no. 11, pp. 131–133, 2001.

[4] M. Usman, E. Mendes, F. Weidt, and R. Britto, "Effort estimation in agile software development: a systematic literature review," in *Proceedings of the 10th international conference on predictive models in software engineering*. ACM, 2014, pp. 82–91.

[5] H. Sharp and H. Robinson, "Three 'c's of agile practice: collaboration, co-ordination and communication," in *Agile software development*. Springer, 2010, pp. 61–85.

[6] S. C. Misra, V. Kumar, and U. Kumar, "Identifying some important success factors in adopting agile software development practices," *Journal of Systems and Software*, vol. 82, no. 11, pp. 1869–1890, 2009.

[7] M. Lindvall, V. Basili, B. Boehm, P. Costa, K. Dangle, F. Shull, R. Tesoriero, L. Williams, and M. Zelkowitz, "Empirical findings in agile methods," in *Conference on extreme programming and agile methods*. Springer, 2002, pp. 197–207.

[8] N. B. Moe, T. Dingsøyr, and T. Dybå, "Overcoming barriers to self-management in software teams," *IEEE software*, vol. 26, no. 6, pp. 20–26, 2009.

[9] N. B. Moe, T. Dingsøyr, and T. Dybå, "A teamwork model for understanding an agile team: A case study of a scrum project," *Information and Software Technology*, vol. 52, no. 5, pp. 480–491, 2010.

[10] M. Hoegl and H. G. Gemuenden, "Teamwork quality and the success of innovative projects: A theoretical concept and empirical evidence," *Organization science*, vol. 12, no. 4, pp. 435–449, 2001.

[11] M. Cohn, *Agile estimating and planning*. Pearson Education, 2005.

[12] E. Letier, D. Stefan, and E. T. Barr, "Uncertainty, risk, and information value in software requirements and architecture," in *Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 883–894.

[13] H. R. Joseph, "Poster: Software development risk management: using machine learning for generating risk prompts," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 2. IEEE, 2015, pp. 833–834.

[14] M. Choetkiertikul, H. K. Dam, T. Tran, T. Pham, A. Ghose, and T. Menzies, "A deep learning model for estimating story points," *IEEE Transactions on Software Engineering*, vol. 45, no. 7, pp. 637–656, 2018.

[15] S. Porru, A. Murgia, S. Demeyer, M. Marchesi, and R. Tonelli, "Estimating story points from issue reports," in *Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering*, 2016, pp. 1–10.

[16] E. Scott and D. Pfahl, "Using developers' features to estimate story points," in *Proceedings of the 2018 International Conference on Software and System Process*, 2018, pp. 106–110.

[17] M. Choetkiertikul, H. K. Dam, T. Tran, and A. Ghose, "Predicting delays in software projects using networked classification (t)," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2015, pp. 353–364.

[18] M. Choetkiertikul, H. K. Dam, T. Tran, and A. Ghose, "Predicting the delay of issues with due dates in software projects," *Empirical Software Engineering*, vol. 22, no. 3, pp. 1223–1263, 2017.

[19] H. Zhang, L. Gong, and S. Versteeg, "Predicting bug-fixing time: an empirical study of commercial software projects," in *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 2013, pp. 1042–1051.

[20] T. L. Dickinson and R. M. McIntyre, "A conceptual framework for teamwork measurement," *Team performance assessment and measurement*, pp. 19–43, 1997.

[21] Y. Lindsjørn, D. I. Sjøberg, T. Dingsøyr, G. R. Bergersen, and T. Dybå, "Teamwork quality and project success in software development: A survey of agile development teams," *Journal of Systems and Software*, vol. 122, pp. 274–286, 2016.

[22] M. Bloch, S. Blumberg, and J. Laartz, "Delivering large-scale it projects on time, on budget, and on value," https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/delivering-large-scale-it-projects-on-time-on-budget-and-on-value, accessed: 2021-03-28.

[23] H. F. Cervone, "Understanding agile project management methods using scrum," *OCLC Systems & Services: International digital library perspectives*, 2011.

[24] K. Schwaber and M. Beedle, *Agile software development with Scrum*. Prentice Hall Upper Saddle River, 2002, vol. 1.

[25] ServiceNow. (2021) Servicenow: Workflows for the modern enterprise. [Online]. Available: https://servicenow.com

[26] M. Cohn, *User stories applied: For agile software development*. Addison-Wesley Professional, 2004.

[27] M. Usman, E. Mendes, and J. Börstler, "Effort estimation in agile software development: a survey on the state of the practice," in *Proceedings of the 19th international conference on Evaluation and Assessment in Software Engineering*, 2015, pp. 1–10.

[28] J. Grenning, "Planning poker or how to avoid analysis paralysis while release planning," *Hawthorn Woods: Renaissance Software Consulting*, vol. 3, pp. 22–23, 2002.

[29] H. Kniberg and A. Ivarsson, "Scaling agile@ spotify," *online], UCVOF, ucvox. files. wordpress. com/2012/11/113617905-scaling-Agile-spotify-11. pdf*, 2012.

[30] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[31] J. Noll and S. Beecham, "Measuring global distance: A survey of distance factors and interventions," in *International Conference on Software Process Improvement and Capability Determination*. Springer, 2016, pp. 227–240.

[32] W. W. Daniel *et al.*, "Applied nonparametric statistics," 1990.

[33] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian journal of statistics*, pp. 65–70, 1979.

[34] M. Choetkiertikul, H. K. Dam, T. Tran, and A. Ghose, "Characterization and prediction of issue-related risks in software projects," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. IEEE, 2015, pp. 280–291.

[35] B. Boehm and R. Turner, "Using risk to balance agile and plan-driven methods," *Computer*, vol. 36, no. 6, pp. 57–66, 2003.

[36] C. d. O. Melo, D. S. Cruzes, F. Kon, and R. Conradi, "Interpretative case studies on agile team productivity and management," *Information and Software Technology*, vol. 55, no. 2, pp. 412–427, 2013.

[37] V. Lalsing, S. Kishnah, and S. Pudaruth, "People factors in agile software development and project management," *International Journal of Software Engineering & Applications*, vol. 3, no. 1, p. 117, 2012.

[38] R. Popli and N. Chauhan, "Agile estimation using people and project related factors," in *2014 International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE, 2014, pp. 564–569.

[39] S. Dorairaj, J. Noble, and P. Malik, "Understanding team dynamics in distributed agile software development," in *International conference on agile software development*. Springer, 2012, pp. 47–61.

[40] J. A. Espinosa, S. A. Slaughter, R. E. Kraut, and J. D. Herbsleb, "Familiarity, complexity, and team performance in geographically distributed software development," *Organization science*, vol. 18, no. 4, pp. 613–630, 2007.

[41] T. Tan, Q. Li, B. Boehm, Y. Yang, M. He, and R. Moazeni, "Productivity trends in incremental and iterative software development," in *2009 3rd International Symposium on Empirical Software Engineering and Measurement*. IEEE, 2009, pp. 1–10.

[42] K. D. Maxwell and P. Forselius, "Benchmarking software development productivity," *Ieee Software*, vol. 17, no. 1, pp. 80–88, 2000.

[43] S. E. Dreyfus and H. L. Dreyfus, "A five-stage model of the mental activities involved in directed skill acquisition," California Univ Berkeley Operations Research Center, Tech. Rep., 1980.

[44] R. W. Zmud, "Management of large software development efforts," *MIS quarterly*, pp. 45–55, 1980.

[45] L. Wallace, M. Keil, and A. Rai, "Understanding software project risk: a cluster analysis," *Information & management*, vol. 42, no. 1, pp. 115–125, 2004.

[46] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, vol. 242, no. 1. Citeseer, 2003, pp. 29–48.

[47] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford *et al.*, "Okapi at trec-3," *Nist Special Publication Sp*, vol. 109, p. 109, 1995.

[48] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*. PMLR, 2014, pp. 1188–1196.

[49] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[50] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[51] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[52] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[53] P. Domingos and M. Pazzani, "On the optimality of the simple bayesian classifier under zero-one loss," *Machine learning*, vol. 29, no. 2, pp. 103–130, 1997.

[54] J. Huang and C. X. Ling, "Using auc and accuracy in evaluating learning algorithms," *IEEE Transactions on knowledge and Data Engineering*, vol. 17, no. 3, pp. 299–310, 2005.

[55] A. Arcuri and L. Briand, "A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering," *Software Testing, Verification and Reliability*, vol. 24, no. 3, pp. 219–250, 2014.

[56] F. Sarro, A. Petrozziello, and M. Harman, "Multi-objective software effort estimation," in *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*. IEEE, 2016, pp. 619–630.

[57] R. Genuer, J.-M. Poggi, and C. Tuleau-Malot, "Variable selection using random forests," *Pattern recognition letters*, vol. 31, no. 14, pp. 2225–2236, 2010.

[58] F. Kortum, J. Klünder, and K. Schneider, "Behavior-driven dynamics in agile development: The effect of fast feedback on teams," in *2019 IEEE/ACM International Conference on Software and System Processes (ICSSP)*. IEEE, 2019, pp. 34–43.

[59] V. R. Basili and R. W. Reiter Jr, "An investigation of human factors in software development." *IEEE Computer*, vol. 12, no. 12, pp. 21–38, 1979.

[60] F. Kortum, J. Klünder, and K. Schneider, "Don't underestimate the human factors! exploring team communication effects," in *International conference on product-focused software process improvement*. Springer, 2017, pp. 457–469.

[61] R. S. Huckman, B. R. Staats, and D. M. Upton, "Team familiarity, role experience, and performance: Evidence from indian software services," *Management science*, vol. 55, no. 1, pp. 85–100, 2009.

[62] R. Schmidt, K. Lyytinen, M. Keil, and P. Cule, "Identifying software project risks: An international delphi study," *Journal of management information systems*, vol. 17, no. 4, pp. 5–36, 2001.

[63] K. Ewusi-Mensah, *Software development failures*. Mit Press, 2003.

[64] S. L. Jarvenpaa and B. Ives, "Executive involvement and participation in the management of information technology," *MIS quarterly*, pp. 205–227, 1991.

[65] P. Ralph and E. Tempero, "Construct validity in software engineering research and software metrics," in *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018*, 2018, pp. 13–23.

[66] A. Arcuri and L. Briand, "A practical guide for using statistical tests to assess randomized algorithms in software engineering," in *2011 33rd International Conference on Software Engineering (ICSE)*. IEEE, 2011, pp. 1–10.

[67] E. Kocaguneli, T. Menzies, and J. W. Keung, "On the value of ensemble effort estimation," *IEEE Transactions on Software Engineering*, vol. 38, no. 6, pp. 1403–1416, 2011.

[68] T. Menzies, Z. Chen, J. Hihn, and K. Lum, "Selecting best practices for effort estimation," *IEEE Transactions on Software Engineering*, vol. 32, no. 11, pp. 883–895, 2006.

[69] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[70] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Information and software technology*, vol. 50, no. 9-10, pp. 833–859, 2008.

[71] T. Dingsøyr, S. Nerur, V. Balijepally, and N. B. Moe, "A decade of agile methodologies: Towards explaining agile software development," 2012.

[72] M. Jørgensen, "A review of studies on expert estimation of software development effort," *Journal of Systems and Software*, vol. 70, no. 1-2, pp. 37–60, 2004.

[73] L. D. Panjer, "Predicting eclipse bug lifetimes," in *Fourth International Workshop on Mining Software Repositories (MSR'07: ICSE Workshops 2007)*. IEEE, 2007, pp. 29–29.

[74] P. Bhattacharya and I. Neamtiu, "Bug-fix time prediction models: can we do better?" in *Proceedings of the 8th Working Conference on Mining Software Repositories*, 2011, pp. 207–210.

[75] E. Giger, M. Pinzger, and H. Gall, "Predicting the fix time of bugs," in *Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering*, 2010, pp. 52–56.

[76] M. Choetkiertikul, H. K. Dam, T. Tran, A. Ghose, and J. Grundy, "Predicting delivery capability in iterative software development," *IEEE Transactions on Software Engineering*, vol. 44, no. 6, pp. 551–573, 2017.

[77] P. Abrahamsson, R. Moser, W. Pedrycz, A. Sillitti, and G. Succi, "Effort prediction in iterative software development processes–incremental versus global prediction models," in *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*. IEEE, 2007, pp. 344–353.