Combining Data Analytics and Developers Feedback for Identifying Reasons of Inaccurate Estimations in Agile Software Development

Marco Conoscenti[a,*], Veronika Besner[b], Antonio Vetrò[a], Daniel Méndez Fernández[b]

[a]*Nexa Center for Internet & Society, DAUIN, Politecnico di Torino*
[b]*Technical University of Munich*

Abstract

[**Background**] Effort estimations are critical tasks greatly influencing the accomplishment of software projects. Despite their recognized relevance, little is yet known what indicators for inaccurate estimations exist, and which are the reasons of inaccurate estimations. [**Aims**] In this manuscript, we aim at contributing to this existing gap. To this end, we implemented a tool that combines data analytics and developers' feedback, and we employed that tool in a study. In that study, we explored the most common reasons of inaccurate user story estimations and the possible indicators of inaccurate estimations. [**Method**] We relied on a mixed method approach used to study reasons and indicators for the identification and prediction of inaccurate estimations in practical agile software development contexts. [**Results**] Our results add to the existing body of knowledge in multiple ways. We elaborate causes for inaccurate estimations going beyond the borders of existing literature; for instance, we show that lack of developers' experience is the most common reason of inaccurate estimations. Further, our results suggest, for example, that the higher the complexity, the higher the uncertainty in the estimation. [**Conclusions**] Overall, our results strengthen our confidence in the usefulness of using data analytics with human-in-the-loop mechanisms to improve effort estimations.

Keywords: `Data analytics; empirical software engineering; agile methods; estimations; mixed methods`

1. Introduction

Project planning and effort estimations required to release a software product are both key aspects for the success of a software project [1, 2, 3]. In the case of agile software development, and specifically of Scrum (the most used agile methodology), effort of user stories is estimated (see Section 2).

User stories estimations can incur in two types of erroneous predictions: over-estimation, where the estimated effort is higher than the actual effort needed, and under-estimation, where the estimation is lower than the actual work needed. Over-estimations might lead to a reduced productivity rate [4], since the development team expands "the work so as to fill the time available for its completion" [5]. Furthermore, over-estimations may result in the belief that the development of a feature or a product in general is not beneficial and, as a consequence, it will be erroneously rejected. Another risk of over-estimations is that new opportunities to undertake new projects might be overlooked due to the belief that there is no capacity available [6]. Whereas accurate estimations (or slightly too low estimations [4]) are known to potentially increase the productivity, large under-estimations are known to cause a low quality of the product due to the resulting time pressure [7]. Moreover, under-estimations might have detrimental effects on the project management, e.g. an inaccurate plan of the staffing of a development team or an inaccurate allocation of

*Corresponding author

Email addresses: `marco.conoscenti@polito.it` (Marco Conoscenti), `veronika.besner@gmail.com` (Veronika Besner), `antonio.vetro@polito.it` (Antonio Vetrò), `daniel.mendez@tum.de` (Daniel Méndez Fernández)

resources. These effects are often hard to handle; for instance, it may take time and skills to render the added staff beneficial and productive [6].

Despite its importance, a measurement-based approach for the improvement of the estimation process is neither a common practice in industry nor is it a practice widely and thoroughly investigated in academic research: in the 9th Annual State of Agile Survey conducted by VersiOne [8], it was shown that, among the 70% of the respondents that use Scrum or one of its variations for agile software development, only 19% of them use a metric that tracks estimation accuracy; moreover, a systematic literature review [9] showed that out of 304 relevant papers published in 76 journals, only 8 papers focused on the effort estimation process and were supported by case studies.

We addressed this issue in a recent work [10] where we combined software project data analytics with elicitation of software development teams' feedback, and based on the validated results, we revised the estimation process and tracked improvements (motivations and foundations of that work are introduced in prior work [11]).

In the study we report on in the paper at hands, we engineered the mixed analysis method experimented in the previous study, and we built an interactive visualization tool which is able to show actual and estimated efforts of the user stories, display the inaccurate ones and it allows developers to leave feedback on the reasons that caused inaccurate estimations. That way, we were able to identify the most common reasons of inaccurate estimations. In addition, by combining the developers feedback with user stories data, we found possible indicators of inaccurate estimations. The study was conducted at a German company which develops web applications with agile methodologies: we analyzed data from three projects.

Our main contributions presented in this manuscript are:

1. An interactive visualization tool which combines software project data analytics with a developer team's feedback to detect inaccurate estimations and identify their most common reasons.
2. Identification of the most common reasons of inaccurate estimates. Relying on the developers' feedback collected in the visualization tool, the most common reason of inaccurate estimations was lack of experience of the developers involved in the user story.
3. Identification of possible indicators of inaccurate estimations. Two of these indicators are: (i) user story size, in particular, the greater the user story is, the higher the probability to inaccurately estimate it, and (ii) the presence of links of the user story to other user stories. To understand whether the found indicators may be reliable indicators of inaccurate estimations, we implemented a naive Bayes classifier which uses the indicators to predict inaccurate estimations.

The paper is structured as follows. In Section 2, we provide the background of our research. Section 3 presents goal and research questions of our study. In Section 4, we provide details on the German company in which the study is conducted. In Section 5, we present the interactive visualization tool we developed. In Section 6, we present the implementation of the Bayes classifier. In Section 7, we show the study results and discuss the limitations in Section 8. Finally, we conclude our results in Section 9.

2. Background

In this Section, we provide an overview of the most important concepts on which we rely in this manuscript.

2.1. Agile Software Development and Scrum

Agile software development is an iterative and incremental development approach. At the end of each iteration, a functional subset of the software product is fully tested and released. Hence, all steps of the development process (requirement analysis, planning, design, implementation, testing and deployment) are performed in each iteration to develop a subset of the software.

As of today, *Scrum* is the most adopted agile methodology [8]. In Scrum, iterations are performed in time-boxed *sprints*, typically encompassing one to four weeks. In Scrum, the role of a Product Owner defines the responsibility for identifying product features, translating them into a prioritized feature list and continually re-prioritizing and refining this list as user needs, business or technical constraints evolve. The

Development Team deals with the actual development of the product via shared functions and responsibilities (e.g. implementation as well as quality assurance). The Scrum Master does whatever is in her power to help the team to be successful and ensures that it stays productive and learning.

2.2. User Stories

In Scrum, a *user story* is a description of a functionality of the software to be implemented, usually required by the user.

The format of user stories is "concise, clear descriptions of the functionality in terms of its value to the end user of the product"[12]. While there is no fixed standard for the syntax of user stories, one common technique is the following template: "As a *[type of user]*, I want *[some goal]* so that *[some reason]*" [13]. The main benefits of defining a user story in this form is that it helps understanding the user's position and avoid developing a functionality which does not reflect the expectations of the user. In this paper, we will refer to a slight variation: "As a *[role]* I want to *[activity]* so that *[business value]*", which better reflects the study context (a company) by focusing on business value.

In each sprint, one or more user stories are implemented [1]. Some user stories are too large or come with too much uncertainty to be implemented in a sprint. These stories are called *epics* and usually are divided into smaller user stories.

2.3. User Story Estimation

A common practice in the planning phase of each sprint is that the effort to realize each user story of that sprint is estimated. Please note that some companies, instead, estimate at task level, i.e. at a lower level than that of a user stories which may have been estimated upfront, e.g., when the project starts. Ideally, however, the estimation covers all parts required to complete the estimated story; that is, not only the implementation effort, but also the testing and integration effort are considered.

At the company in which we conducted the study at hands, two estimation units are used: story points and t-shirt sizes. Story points describe the size of a user story with a number belonging to the Fibonacci series. T-shirt sizes [10] use the known common categories S, M, L, XL, and XXL. Both estimation units are relative, i.e. one effort category is twice as much than the previous one. For example, the estimated effort for a story with two points assigned should be twice as much than for a story with one point assigned.

3. Goal, Research Questions and Metrics

The goal of the present study is to identify the most common reasons of inaccurate estimations of user stories and the factors which likely produces inaccurate estimates. The study is conducted by analyzing three projects developed at a German company which produces web applications (cf. Section 4 for more details on the projects and the company).

To accomplish the goal, we developed an interactive visualization tool (details in Section 5). The tool shows actual and estimated effort of user stories. The user stories are divided into categories, according to the estimated effort required for implementing the stories. Then, using a threshold based on the mean and standard deviation of estimates in each category, we highlight the *potential* inaccurately estimated user stories for each category. Such categorization is validated directly by the developers relying on a functionality that allows them to confirm the classification and, in addition, provide what they think were the reasons for such inaccurate estimates (both on a closed list and on a free text box).

In the following, we introduce the research questions we formulated starting from the research goal.

RQ1: Which metric is able to capture inaccurately estimated user stories?

We want to determine which metric correctly detects inaccurate estimations. To achieve this, we validate the correctness of the metric used in the tool with feedback that the developers left while using the interactive visualization.

[1]It can also be zero, when other task are completed - such as refactoring

Metric. To answer the question, we used as metric in the visualization tool mean and standard deviation to detect inaccurate estimations (a user story is considered as inaccurately estimated if its implementation effort is one standard deviation higher/lower than the mean of the efforts of the other user stories in the same category). We then compared the results obtained with those obtained used a threshold based on median an interquartile range ,which were used in a previous work within the same company [10].

RQ2: What are the most common reasons for under- and over-estimations?

We collect the most common reasons for under and for over-estimations in the tool by providing the developers a check box list with common reasons as well as a free text field for letting them explain additional reasons. We analyze the findings and derive the most common reasons for inaccurate estimations in the three projects involved in the study.

Metric. We used as metric the number of occurrences of reasons of inaccurate estimations provided by the developers.

RQ3: What are the possible indicators of inaccurate estimates?

We want to analyze whether there exists certain data that is known before a story is estimated and implemented, which serves as indicator for a possible inaccurate estimation or increases/decreases the chance that a story is inaccurately estimated. We identified several possible indicators, which we call herein "influence factors", that we check for correlations with inaccurate estimations. In the following, the possible influence factors we analyzed in this paper are listed:

- *Epic of the user story:* Most stories do belong to a certain epic. In the context of the study context, an epic typically represents the implementation category the story belongs to. For example, a story can belong to the epic "Search", or "Product Details". Each epic consists of multiple different stories and most epics occur in multiple projects in the same way. We investigate whether the inaccurate estimates are clustered with respect to the related functionality, i.e. the epic. In fact, in a previous study [11], we collected initial evidence on recurring topics for user stories which correlated with inaccurate estimations and bugs in the corresponding code.
- *Presence of User Acceptance Criterion (UAC):* To be able to quantify when a story is done, every story should have a user acceptance criterion, which consists of measurable and testable criteria that define when a story is completed. In practice, this criterion is sometimes missing. The underlying hypothesis is that having a UAC makes the estimation easier and thus more accurate.
- *Usage of the standard user story form*: We refer to that form as a textual description following the pattern "As a *[role]* I want to *[activity]* so that *[business value]*". In the study, we want to examine whether the probability that a story is inaccurately estimated is correlated with the usage (or not)of that standard form.
- *Links to other user stories:* Some stories are linked to several other stories or to other tasks. This happens if they depend on each other or regard shared functionalities (again, refer to [11]). We want to examine whether stories that are linked to other stories have a higher probability to be inaccurately estimated than stories that are independent from other stories.
- *User story size:* We want to examine whether the size of a story influence the estimation, and in particular, whether large user stories are more difficult to estimate than small ones.

Metric. To answer the research question, we used as metric the Pearson correlation coefficient [14] computed between inaccurate estimations (identified by the developers) and the aforementioned supposed influence factors.

4. Study Context

This study was conducted at a medium-sized German company consisting of approximately 70 developers that develops web applications applying agile techniques. The company is specialized in developing e-commerce systems by using the two popular content management systems TYPO3[2] and Magento[3]. They

[2]https://typo3.org/
[3]https://magento.com/

have more than 17 years of experience in developing web applications: they developed more than 100 TYPO3 and more than 40 Magento projects and are a certified partner for both frameworks.

The company started to use agile methodologies in 2011 and is now structured into agile teams, each consisting of several developers, a Scrum Master, and a Product Owner. These teams work mainly self-organized and can also decide if they prefer working with Scrum or other agile methods like Kanban. As the purpose of the study is to compare estimated effort with actual effort, we focused on the projects using Scrum because we need projects where the teams perform estimations.

The company uses the project management software JIRA[4] and the extension for agile environment "JIRA agile" to plan its projects, manage the work, and log the time the developers work on certain tasks. If a team develops with Scrum, they can log the story point values they estimated in a sprint planning meeting directly to the corresponding user stories in the tool. All the relevant data needed to compare the estimated effort of one user story with the actually needed implementation effort for that story can therefore be retrieved directly from JIRA. Furthermore, the interactive visualization we developed is embedded directly into JIRA to enable convenient usability and real-time adjustments.

The present study was mainly conducted in a secondary office of the company, where one team with seven members is located; therefore, most of the research was done in collaboration with this team. The study analyzes three projects developed by the company. The details of the projects are described in the following.

4.1. Project 1

In Project 1, an online shop was developed with Magento. The developer team consisted of three back-end developers, an external front-end developer, a part-time working student, a Product Owner, and a Scrum Master; however, only two back-end developers were present most of the time. The team developed with Scrum in weekly iterations and used T-shirt sizes for estimating the user stories. This is the only project where the authors of this study accompanied the Scrum team during the complete course of the project and the questions about inaccurate estimations were mainly answered directly after implementation. Stories in this project were completed in two months, and we analyzed all of them: in total we looked at 43 estimated and completed user stories.

4.2. Project 2

Project 2 was developed by the same developers as Project 1 and additionally by an internal front-end developer, who left during the course of the project. Also for Project 2 T-shirt size were used a estimation unit. The project took over a year, but we analyzed user stories taken in a time frame of three months at the middle of the project. We did not use stories preceding that period because the team did not remember many details about the implementation of user stories.. In total, we looked at 30 stories of this project.

4.3. Project 3

Project 3 was based on TYPO3. The team consisted of four developers and the project was conducted with Scrum and a variation of it. We analyzed user stories collected during three months at the middle of the project, in total 51 user stories. The main difference with respect the previous two projects is the usage of story points as estimation unit in Project 3. As our tool has been programmed to accept all kinds of estimation values this was not a problem, but it has to be considered when comparing the results of the analysis of the different projects.

5. The Interactive Visualization Tool

We developed an interactive visualization tool to show actual and estimated efforts for the user stories in the analyzed projects and that suggests which stories were inaccurately estimated. Such stories are then validated by the developers themselves, who can also indicate the motivations for under and over-estimations.

[4]https://www.atlassian.com/software/jira

5.1. The Visualization

The integration into the JIRA environment was a requirement for the tool. The reasons were that developers can use it locally (as long as there is an internet connection) while ensuring access control via the JIRA environment, thus, protecting project-specific sensible information. Further, all links to user stories work without an additional authorization process, the visualization works with real-time data, which does not need to be exported first, and if anyone makes a change, it is displayed right away when refreshing the page.

We have chosen to implement the visualization as a gadget, which can be placed on any dashboard in JIRA. The advantage we see in this solution is that we perceive it as flexible to use: one can, for example, choose to show the gadget on the whole screen to get a good overview or one can use only half of the screen for the visualization gadget. Then, one can put another gadget with informative information on the other half of the dashboard to be able to compare data of the interactive visualization with other data or documents. The gadget is embedded in a JIRA plugin. This makes it possible that every JIRA user in the company can easily include the gadget to his or her dashboard by selecting it from an available gadgets list.

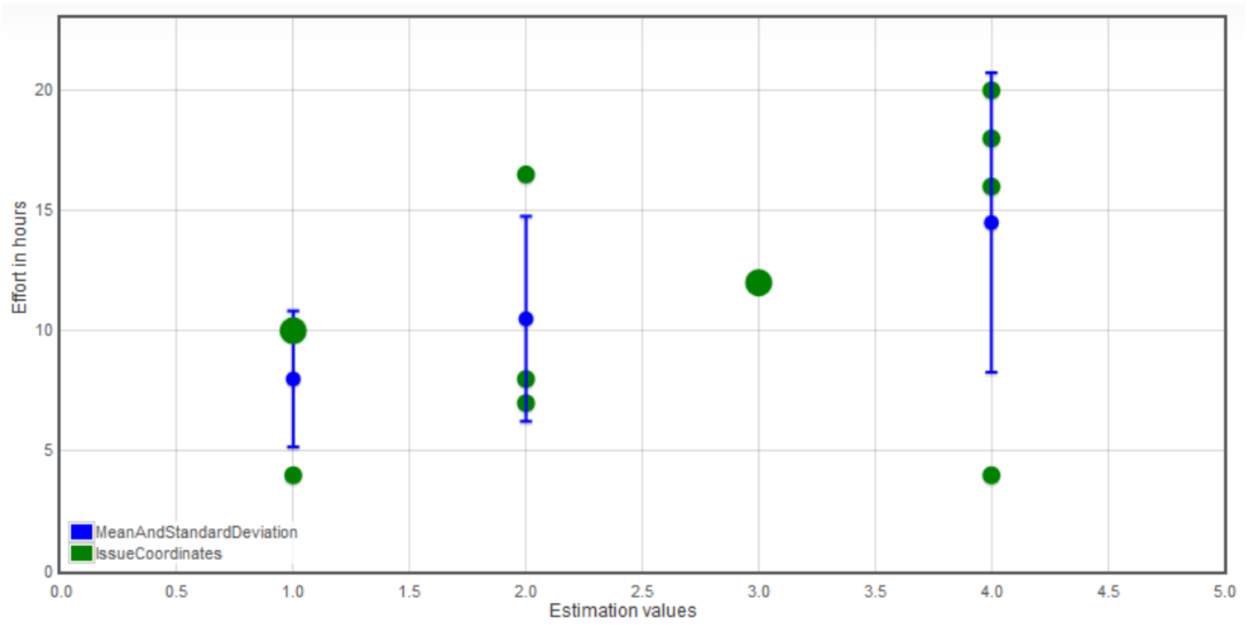

Figure 1: The visualization: actual and estimated effort of user stories.

In Figure 1, we show such an exemplary visualization. Each small green point represents a single user story and the larger green points represent clusters of two or more user stories. The x-axis values correspond to the estimation values of each user story. In our case, the estimation values 1, 2, 3, 4, 5 correspond to each T-shirt size estimation category S, M, L, XL, XXL, because JIRA only supports estimating with numerical values. The y-axis represents the actually needed effort to complete each user story, expressed in hours. This effort combines all kinds of effort booked on that story (implementation, testing, etc.). The coordinates of each green point on the grid therefore represents the story estimation value in relation to the actual effort needed to complete that story.

In addition, for each of the five estimation categories, we show mean and standard deviation of the implementation effort of the user stories in that category. The blue points represent the means and the bars are error bars that use standard deviation. Mean $\overline{E}$ and standard deviation σ_E of the implementation effort are calculated using the following formulas:

$$\overline{E} = \frac{1}{n}\sum_{i=1}^{n} e_i \qquad (1)$$

$$\sigma_E = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(e_i - \overline{E})^2} \tag{2}$$

where e_i is the actual implementation effort of the i-th user story in a certain estimation category, for $i = 1, 2, ..., n$, and n is the total number of user stories in a certain estimation category.

A user story is considered as inaccurately estimated if its implementation effort is one standard deviation higher/lower than the mean of the efforts of the other user stories in the same category. This threshold was chosen following a conservative line, starting from the results of our previous study on estimation issues with the same company [10]. In other words, given that every story tagged as inaccurate would be validated by one or more developers, we preferred to have more false positive rather than getting false negatives and, thus, missing useful data for our analysis. In a previous work within the same company [10], we used median and interquartile range (IQR): such metric is not shown in the tool, but we made a comparison between the two metrics as presented in Section 7.1.

Finally, the inaccurately estimated user stories are shown as green points which lie outside the blue bar.

5.2. Feedback Collection

The tool is able to collect developers' feedback on user stories. In particular, we collect data about the estimation with two different questions. These two questions are applicable for every user story that has been estimated. At first, we want to know if the developer thinks that a certain story was inaccurately estimated and in that case, if the estimation was an under-estimation or an over-estimation. Secondly, if the story was inaccurately estimated, we ask to the developers about the reasons: we provide on a check-box list the most common reasons retrieved from the literature ([15, 16, 17, 18, 19, 20]), together with an "Other" check-box and a comment field, where to add explanations and also mention additional reasons. We provide the following most common reasons of inaccurate estimations among which the developers can choose:

- **Understandability problems of the story:** Due to understandability problems or unclear definition of user acceptance criteria, the developers thought the story would need more or less time to be implemented. During the implementation it became clear that the desired functionality required less or more time than first thought.
- **Lack of experience:** The story was inaccurately estimated because the estimating developers or the developers involved in the implementation lacked experience.
- **Inaccurately assumed (non-)standard functionality:**
  - **In the over-estimation case:** The developers thought during the estimation process they would have to implement the feature themselves, while there was a standard functionality available (for example in Magento or TYPO3) that provided exactly that feature. Therefore it took much less time to implement the feature than expected.
  - **In the under-estimation case:** The developers thought that the desired feature could be easily resolved with a standard functionality provided by Magento or TYPO3 (or a previously developed project), but that functionality was not usable. Therefore implementation took longer than expected.
- **Error with effort/estimation recording:** The story was inaccurately estimated, because some error in the effort or estimation logging occurred; for example, because developing time was erroneously booked on that user story.
- **Rejection of the story:** Rejection of a user story can either mean that the story did not pass the review of a second developer, or that it did not pass the review of the product owner or client, who then rejects the completion of that story. Therefore, the user story must be improved and this can cause the whole implementation to take longer than expected.
- **Technical problems:** While implementing a story, technical problems occurred that had to be resolved and were booked on that story; for example, a code refactoring activity, a maintenance task on the servers, or an adjustment of the testing system was necessary. This category captures any problems at the technical level not directly related to the functionality of the story.

- **Unnoticed side task:** Some other feature, which was not taken into consideration when estimating the story, was implemented with the story. This may cause a greater effort than the estimated one.

6. The Bayes Classifier

We implemented a naive Bayes classifier which uses influence factors (cf. Section 3) to predict whether a user story is likely to be inaccurately estimated. The classifier is intended as a mean to test whether the factors that are mostly related to inaccurate estimations (identified in Section 7.3) may be possible reliable indicators of inaccurate estimations.

To create the Bayes classifier we used the data mining tool RapidMiner[5]. In Figure 2, we depict the process that creates the classifier.

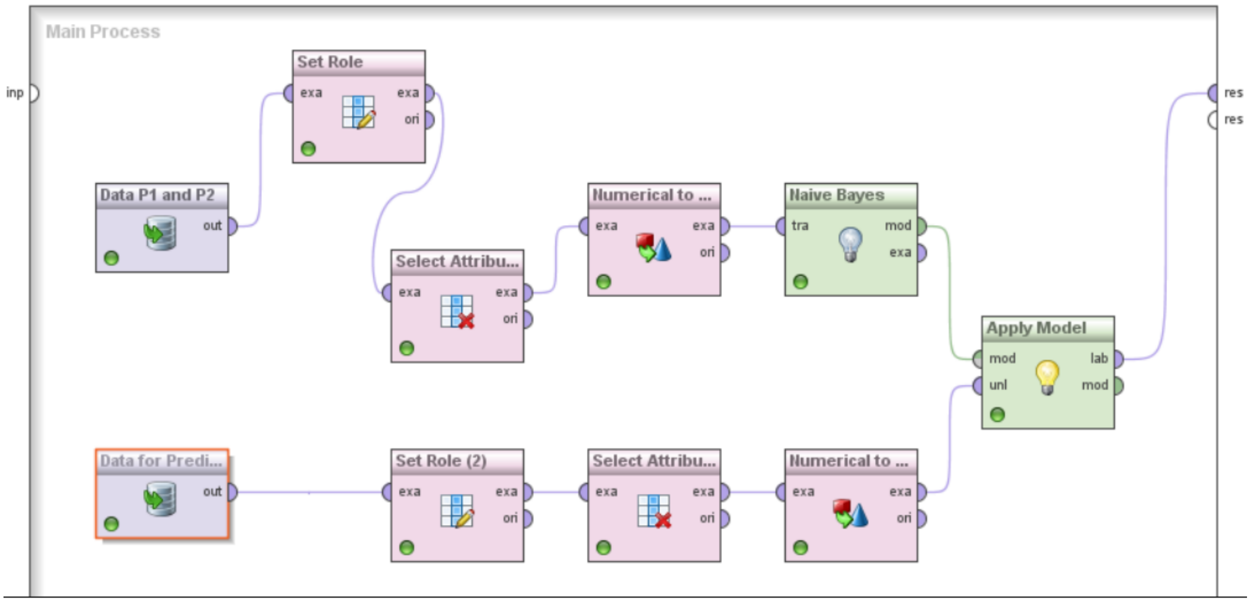

Figure 2: Implementation of a naive Bayes classifier to predict inaccurate estimations.

Only the user stories from Project 1 and Project 2 are used in this part, due to low number of data on the possible influence factors found in Project 3. From the stories in these projects, the information provided by the developers whether the story were inaccurately estimated or not is used to label the data and to estimate the prediction accuracy of the classifier.

With a training set built in such way, we apply the "Naive Bayes" operator provided by RapidMiner to produce the prediction given an input set of user stories to be predicted. Results on the prediction accuracy of the classifier using different sets of influence factors in the prediction mechanism are presented and discussed in Section 7.3.1.

7. Results and Discussion

In the following, we present the results of our study, structured according to our research questions.

[5]The RapidMiner tool is available at https://rapidminer.com/

7.1. RQ1: Which metric is able to capture inaccurately estimated user stories?

We asked the developers to mark which stories are inaccurately estimated according to them. That way, we can determine whether the metric used by the visualization tool is able to correctly identify inaccurate estimations, and we compare it with another metric used in previous work, based on median and interquartile range.

We provide as descriptive statistic in Table 1 the mean and median of the implementation effort in each estimation category for the three projects, where we observe that the mean tends to be higher than the median.

Metric	**Estimation category**				
	1	**2**	**3**	**4**	**5**
Mean	4.38	9.25	18.7	28.63	-
Median	3.5	8.6	18.25	25.75	-

(a) Project 1.

Metric	**Estimation category**				
	1	**2**	**3**	**4**	**5**
Mean	2.32	5.68	12.92	36.75	22.08
Median	1.5	4.5	15	36.75	22.5

(b) Project 2.

Metric	**Estimation category**								
	0.25	**0.5**	**1**	**2**	**3**	**5**	**8**	**13**	**20**
Mean	13	2.75	3.84	9.5	15	28.31	20.29	40.88	48.25
Median	13	2.75	3.5	8.5	13.75	22	18.75	40.86	48.25

(c) Project 3.

Table 1: Mean and median values of the implementation effort for each estimation category.

Table 2 shows the results on the inaccurate estimations identified when using mean and standard deviation as metric (abbreviated with STD), compared with the metric based on median and interquartile range (abbreviated with IQR), and the ones marked by the developers.

With STD, as explained in Section 5.1, a user story estimation is considered as inaccurate when the actual user story effort is a standard deviation away from the mean effort of the user stories in the same estimation category.

With IRQ, user stories which are 25% above the median and 25% below the median are considered inaccurately estimated[6].

STD and IRQ are compared with respect to the developers' indications. It is important to note that this analysis assumes that the developer feedback is correct. In our opinion, this is a reasonable choice, since they directly worked on the projects and know well the user stories they implemented in the code.

In the case of Project 1, IQR performs slightly better than STD. It correctly detects 16 inaccurate estimations out of the 21 identified by developers, so 2 more with respect to STD. Hence, the accuracy of

[6]The median value separates the upper half of the data points (in our case, the estimation effort) from the bottom half. If the number of points is odd, it will equal the middle point, if the number is even, it will lie right between the two middle points

IQR results 76.2%, with respect to 66.6% of STD. In the case of Project 2, STD outperforms IQR as the latter produces more false positives (i.e., actually correct estimations detected as inaccurate). In the case of Project 3, IQR performs better than STD, as it correctly detects two more inaccurate estimations (18 against 16, out of 22 individuated by the developers). The accuracy of IQR is 81.8%, greater than the one of STD (72.7%).

STD	**IQR**	**Developers**
14	16	21

(a) Project 1.

STD	**IQR**	**Developers**
10	14	11

(b) Project 2.

STD	**IQR**	**Developers**
16	18	22

(c) Project 3.

Table 2: Number of inaccurate estimations identified by: mean and standard deviation (STD); median and interquartile range (IQR); the developers.

7.1.1. Summary and Discussion of the Main Findings

We computed the accuracy [7] of STD and compared it with the accuracy of IQR. Both metrics tend to produce some false negatives (inaccurate estimations not detected) and barely none false positive (correct estimations detected as inaccurate), which only occur in Project 2. For this reason, a solution to improve the accuracy could be to multiply the standard deviation or the interquartile range by a factor lower than 1. By doing so, they will detect more inaccurate estimations.

In addition, we showed that the accuracy of IQR is slightly greater that the one of STD in Project 1 and 3. For this reason, it could be useful to show both metrics in the visualization tool (which currently shows only STD).

7.2. RQ2: Which are the most common reasons for under and over-estimations?

Additionally to the most common reasons we provided in the tool, among which the developers can choose (cf. Section 5.2), the developers added other reasons in the comment field of the tool. We were particularly interested in those reasons as they allow us to investigate causes beyond the boundaries of existing literature. Those additional reasons for inaccurate estimations are:

- **Duplicate/Redundancy:** Sometimes, there exist two or more stories, which concern parts of the same functionality to be implemented. If these stories are not separated carefully enough, it can happen that the developers estimate all stories as if they have to implement that same feature in every single story. When developing them, once the functionality is implemented, the other stories do need much less time to implement than estimated.
- **Complexity of functionality under-estimated:** There are certain software functionalities, for example filtering, which are complex and can often lead to bugs and other difficulties that have to be solved in order to complete the story. If it is neglected in the estimation process, this easily leads to under-estimations.
- **Complexity of functionality over-estimated:** On the contrary to the previous reason, the developers often take into account that a story reflects a difficult functionality when estimating. But if everything works as intended without any occurring problems, the actual effort to implement the story is lower than the estimated one, leading to an overestimated story.

[7] We remind that, here, accuracy is in terms of correspondence with what the developers consider to be a inaccurate estimation and what our accuracy measures categorize it as inaccurate estimation.

- **Gold plating:** *Gold plating* is the situation in which developers add additional features or refine them although they are not required in the user story and may not be absolutely needed. While this may be acceptable as long as only very small features are added or small bugs are solved, this can become problematic if too much time is spent on it.
- **New team member:** If a team member is new in the company or new in the project under consideration, he/she needs introductory training. He/she will not be as efficient as members completely familiar with the project and the used technologies. It is difficult to take this factor into consideration as the estimation process should not be dependent on who actually implements a story. Still, it is a reason why under-estimations can occur. Theoretically, a new team member can also be a reason for overestimating a story, if the team considers the introductory learning factor when estimating but the new member does not need as much time as thought to familiarize himself/herself with the project.
- **External resources needed:** It can occur that a team is depending on some external resources to be able to complete a certain story. Dependence on such an external resource can not only delay the implementation process regarding that story, but also often includes much communication overhead and potential problems that should be taken into account when estimating. If an overestimation occurs within this category, it means that the potential problems of external resources were considered but not observed during the implementation.
- **Involvement of multiple developers on a story:** It happens that a story has to be implemented by several developers in sequential order. In this case, if a developer is not available, the others must re-do their work to be able to complete the story, thus leading to an under-estimation. On the contrary, in the case of pair programming, over-estimations can occur if the two developers work unexpectedly efficiently together.

7.2.1. Occurrences of Reasons for inaccurate Estimations

Reason	**Under-estimations**				**Over-estimations**				**Total**
	P1	**P2**	**P3**	**Tot**	**P1**	**P2**	**P3**	**Tot**	
Understandability problems of the story	1	0	0	1	0	0	1	1	2
Lack of experience	10	1	8	19	4	3	0	7	26
Inaccurately assumed (non-)standard functionality	2	0	0	2	1	1	1	3	5
Error with effort/estimation recording	0	0	1	1	2	1	1	4	5
Rejection of the story	2	1	2	5	-	-	-	-	5
Technical problems	0	0	1	1	-	-	-	-	1
Unnoticed side task	4	0	2	6	-	-	-	-	6
Duplicate/Redundancy	0	0	2	2	2	1	4	7	9
Complexity of functionality under-estimated	1	0	2	3	-	-	-	-	3
Complexity of functionality over-estimated	-	-	-	-	5	5	1	11	11
Gold plating	2	0	2	4	-	-	-	-	4
New team member	1	0	0	1	0	0	0	0	1
External resources needed	1	0	3	4	0	0	0	0	4
Involvement of multiple developers on a story	1	0	0	1	0	0	0	0	1

Table 3: Occurrences of reasons for over-estimations and under-estimations in the three projects (P1, P2, P3) and in total

Tables 3 show the occurrences of the reasons for under and over-estimations for the three projects and in total, collected from the feedback left by the developers in the tool. When a reason can lead only to over-estimation, the column relative to the under-estimation is filled with the symbol "-" (and vice-versa for under-estimation). In addition, it may happen that for a story more reasons have been indicated.

In the case of Project 1, we found in total 39 reasons. The reason that occurred the most was "lack of experience", with 10 under-estimations and 4 over-estimations. The developers used this reason to indicate inexperienced developers as well as inexperienced estimators (it happened often that back-end developers had to estimate front-end stories). Besides, "unnoticed side task" was the reason for most under-estimations and "complexity of functionality over-estimated" the most common reason for over-estimating. The only reason that did not occur was "technical problems". All other reasons only occurred once or twice for under or over-estimations.

In the case of Project 2, the most occurred reason for inaccurate estimations was “complexity of functionality over-estimated”, with 5 over-estimations. “Lack of experience” is responsible for 1 under-estimation and 3 over-estimations. Three other reasons (“inaccurately assumed non-standard functionality”, “error with effort/estimation recording”, “duplicate/redundancy”) caused over-estimations and the “rejection of the story” caused 1 under-estimation. However, we identified only 13 occurrences of reasons, which limits the significance of our findings.

In the case of Project 3, we identified 31 occurrences of reasons. “Lack of experience” is the most occurred reason for inaccurate estimations: it caused 8 under-estimations. “Duplicates/redundancy” occurred 6 times. In the case of over-estimations, it occurred because the main functionality of the user story at issue had already been implemented in another story, thus considerable parts of the story resulted redundant. In the case of under-estimations, a functionality that had to be implemented in more stories was actually implemented in just one, causing the story to be under-estimated. This project was also the only one where external resources were needed and caused 3 under-estimations. Most other reasons occurred once or twice; only the two categories “new team member” and “involvement of multiple developers on a story” did not occur at all.

7.2.2. Summary and Discussion of the Main Findings

Combining the results obtained for the three projects, the most common reason for inaccurate estimations was “lack of experience”, which occurred 26 times in total. This happened because either the estimation or the implementation of the user story was done by inexperienced developers.

The other reason which occurred frequently (11 times) was “complexity of functionality over-estimated”. However, according to developers who has been questioned on the topic, they consider that stories over-estimated for this reason do not constitute a serious problem. They thought that higher implementation effort than the actual one was very likely required, since the functionalities in the over-estimated stories were difficult to implement. The fact that less time was actually needed do not lower the productivity of the team.

Regarding the remaining reasons, they occurred less with respect to the previous ones. Therefore, we can conclude that the best way to reduce inaccurate estimations may be to keep consistency in the development teams and to guarantee that stories are always supervised by experienced team members.

7.3. RQ3: What are the possible indicators of inaccurate estimates?

Table 4 summarizes the user stories attribute used as possible indicators and their data types. Table 5 shows the Pearson correlation coefficient [14] between inaccurately estimated stories and the five indicators selected. The bar charts of the estimations in relation to each indicator, and for each project, are reported in an online appendix[8], and briefly summarized in the following, together with a comment on the correlations showed in Table 5. As far as correlations are concerned, we refer to the classification of Cohen [21]: correlation values greater or equal than 0.5 are seen as strong, values between 0.3 and 0.5 as moderate, and values greater than 0.1 and lower than 0.3 are seen as weak.

Epics. Correlations are weak or absent. However, when looking at the single occurrences validated by the developers, we observe that in every project there are epics that contain more incorrectly estimated stories than correct ones. Therefore, we deem reasonable to consider – although with caution – links to epics as a possible influencing factor.

UAC. There is only one project (Project 2) with a significant number of missing UAC (14): in that project, every under-estimation occurred in a story without UAC. However, such finding is not robust, as only 2 under-estimations occurred in that project. Therefore, we cannot consider UAC as a possible influence factor that the presence or absence of UAC does not greatly impact the correctness of an estimation.

[8]Available at https://researchdata.nexacenter.org/daese/daese_appendix.pdf

Indicator	**Explanation**	**Data Type**
Link	Whether the story has links to other stories	Boolean (“Yes” or “No”)
UAC	Whether the story contains a UAC	Boolean (“Yes” or “No”)
Story form	Whether the story is formulated in the user story form	Boolean (“Yes” or “No”)
Story size	Estimation value of the story	Numerical
Epic	Which epic the story belongs to	Textual
Inaccurate estimation	Whether the story was inaccurately estimated	Boolean (“Yes” or “No”)
Under/over estimation	Whether the story was over/under estimated	Textual (“No”, “Under”, or “Over”)

Table 4: User stories attributes used as indicators in the correlations.

Attribute	**Project 1**		**Project 2**		**Project 3**	
	Inaccurate	**Under/over**	**Inaccurate**	**Under/over**	**Inaccurate**	**Under/over**
Link	0.162	0.051	0.386	0.397	-0.154	-0.077
UAC	0.216	0.194	0.018	0.116	0.162	0.085
Story form	-0.026	0.011	0.196	0.225	0.119	0.034
Story size	0.294	0.397	0.311	0.173	-0.012	0.044
Epic	0.083	0.108	-0.091	-0.133	0.169	0.232

Table 5: Correlation between inaccurate (and under/over) estimations and the possible indicators of inaccurate estimations.

Story form. Regarding the user story form, Project 3 contains only three user stories that are not formulated in the recommended form, the other two projects contain a significant amount of stories where this form is missing. In Project 1, the number of inaccurate estimations of stories in user story form is close to the number of those not formulated in the form (9 vs. 12 respectively). In Project 2, the number of over-estimations is higher for stories in user story form (7 vs. 2) and the two under-estimations are also in the correct form. Therefore we conclude that in our dataset the user story form does not seem to be correlated with inaccurate estimates.

Links to other stories or tasks. Correlations are moderate for Project 2, and low or irrelevant for Project 1 and Project 3. By looking at the occurrences on each project, in Project 1 and Project 2 the stories with links are much more inaccurately estimated than the stories without any link. Out of 38 stories with links, 21 are inaccurately estimated, while out of the 35 stories without links, only 11 are inaccurately estimated. In Project 3, only a few stories have links and, thus, we cannot draw any reasonable consideration on it. Although evidence is not strong, overall, we think that at least in presence of stories with links to other stories or tasks, developers should be more cautious when estimating.

User story size. Correlations are moderate and low for Project 1 and 2. In fact, with the exception of Project 3 where over-estimations are distributed rather equally, over-estimations tend to occur more often for larger stories. This suggests that in our projects, the smaller stories are more correctly estimated, while the larger ones tend to be more inaccurately estimate.

7.3.1. Results of the Bayes Classifier

In this Section, we present the results of the naive Bayes classifier we implemented, which uses influence factors to predict whether a user story is likely to be inaccurately estimated (cf. Section 6). The classifier is a mean to test whether the used influence factors may be reliable indicators of inaccurate estimations.

Results on the model fitting are presented in Table 6, with two runs: the first using all possible influence factors mentioned in Section 3, the second one using only the influence factors identified in this Section as connected to inaccurate estimates.

In the first case, 33 out of 43 user stories in Project 1 were predicted correctly, 23 out of 30 stories in Project 2 were correct. Therefore, we get a prediction accuracy of more than 76% in both projects. In the second case, the prediction accuracy slightly increase and is around 80% in both projects.

In addition, we ran another test to validate the the Bayes classifier itself, in which we use as training set only data from Project 1 and we predict user stories from Project 2. That way, we do not use as training set any data that we want to predict. As influence factors, we considered only the three that are connected to inaccurate estimates and produced higher prediction accuracy in the previous runs. With this setup, we got 19 correct predictions of Project 2, 7 incorrect predictions, and 4 stories that could not be predicted because they contained story point numbers, which did not exist in Project 1. The resulting prediction accuracy is 63%.

Used Influence Factors	**Project**	**Correct predictions**	**Fitness**
Epic, Story size, Link, UAC, Story form	Project 1	33/43	76.7%
Epic, Story size, Link, UAC, Story form	Project 2	23/30	77%
Epic, Story size, Link	Project 1	34/43	79.1%
Epic, Story size, Link	Project 2	24/30	80.0%

Table 6: Prediction accuracy of the Bayes classifier.

7.3.2. Summary and Discussion of the Main Findings

We do not find any strong correlation about indicators and inaccurate estimates, but some moderate relationship with three indicators: user story size, links to other stories, and epics.

Using only the three indicators in the naive Bayes classifier results were slightly better than using the whole set of indicators (around 80% vs. around 76%). This may suggest that the three above-mentioned indicators may be possible indicators of inaccurate estimations.

We do not think that results are strong enough to draw significant conclusions. However, we deem reasonable to conclude that in presence of these three indicators, the developers might use a bit more caution when estimating the effort of an user story.

8. Limitations

One of the main limitation of our work is the fact that we used data only from three projects and one company: as a consequence, generalization of results is an issue.

Another limitation regards the comparison of the metrics used for identifying inaccurate estimations: in fact, the accuracy of the metrics greatly depends on how strict the developers were in marking estimations as inaccurate, since false negatives (inaccurate estimations not detected by the metric) and false positives (correct estimations detected as inaccurate by the tool) are defined by developers. Therefore, it is not possible to compare the two metrics in absolute terms.

9. Conclusions

With our study presented in this manuscript, we contribute to a better understanding and improvement of estimation processes based on user stories. Our overall goal was to identify most common reasons of inaccurate estimations and indicators for inaccurate estimations. The study was conducted using three project environments hosted by a German company which develops web applications with the agile methodology.

To achieve the goal, we followed a research methodology previously experimented within the same company: we combine data analytics with human feedback by means of an interactive visualization. To this end, we developed a tool which shows actual and estimated implementation effort of user stories, and which is able to suggest inaccurately estimated stories following a rather conservative statistical approach. The user can further provide feedback on the reasons of inaccurate estimations for each of the user stories.

In the case of our study, we discovered that the most common reason for inaccurate estimations was the lack of experience of the involved developers.

In addition, combining developers feedback with user stories data, we identified – although without strong statistical evidence – possible indicators of inaccurate estimations: the larger user stories, the user stories which have link to other stories, tasks, or epics are the ones which are most likely to be inaccurately estimated. This seems to indicate that the higher the complexity, the higher the uncertainty in the estimation.

The results support researchers already in gaining a better understanding for how to potentially discover inaccurate estimations and causes that lead to such inaccurate estimations, thus, allowing to steer further research on those results. To further increase the impact for practitioners, we will explore in the future how to best use tools like the one we implemented in context of our study to support detecting and mitigating inaccurate estimations. This shall support transferring our analytical results to conceptual contributions applicable in practice.

Acknowledgements

The authors wish to thank the employees at the German company where this study was conducted. The authors whish to thank also Sebastian Eder and Sacha Storz for the contribution and stimulating discussions as well as Magne Jørgensen for feedback on earlier versions of this manuscript.

References

[1] S. McConnell, Software estimation: demystifying the black art, Microsoft press, 2006.

[2] E. Egorova, M. Torchiano, M. Morisio, C. Wohlin, A. Aurum, R. B. Svensson, Stakeholders' perception of success: An empirical investigation, in: 2009 35th Euromicro Conference on Software Engineering and Advanced Applications, 2009, pp. 210–216. `doi:10.1109/SEAA.2009.33`.

[3] P. Mohagheghi, M. Jørgensen, What contributes to the success of it projects? an empirical study of it projects in the norwegian public sector, Journal of Software 12 (9) (2017) 751–758. `doi:10.17706/jsw.12.9.751-758`.

[4] N. Nan, D. E. Harter, Impact of budget and schedule pressure on software development cycle time and effort, IEEE Transactions on Software Engineering 35 (5) (2009) 624–637. `doi:10.1109/TSE.2009.18`.

[5] C. N. Parkinson, Parkinson's Law, Buccaneer Books, 1996.

[6] N. Mittas, L. Angelis, Overestimation and underestimation of software cost models: Evaluation by visualization, in: 39th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2013, Santander, Spain, September 4-6, 2013, 2013, pp. 317–324.

[7] M. JøRgensen, A review of studies on expert estimation of software development effort, J. Syst. Softw. 70 (1-2) (2004) 37–60. `doi:10.1016/S0164-1212(02)00156-5`.
URL `http://dx.doi.org/10.1016/S0164-1212(02)00156-5`

[8] VersionOne, The 9th annual state of agile survey, (as consulted online on 2015-05-03) (2015).
URL `http://scrumgroup.org/wp-content/uploads/2015/03/state-of-agile-development-survey-ninth.pdf`

[9] M. Jorgensen, M. Shepperd, A systematic review of software development cost estimation studies, Software Engineering, IEEE Transactions on 33 (1) (2007) 33–53.

[10] A. Vetro, R. Dürre, M. Conoscenti, D. M. Fernández, M. Jørgensen, Combining data analytics with team feedback to improve the estimation process in agile software development, Foundations of Computing and Decision Sciences 43 (4) (2018) 305–334.

[11] A. Vetrò, S. Ognawala, D. Méndez Fernández, S. Wagner, Fast feedback cycles in empirical software engineering research, in: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, Vol. 2, 2015, pp. 583–586. `doi:10.1109/ICSE.2015.198`.

[12] J. Sutherland, Scrum handbook, Scrum Training Institute Press, 2010.

[13] M. Cohn, User Stories and User Story Examples by Mike Cohn, [Online; accessed 28. Nov. 2018] (Nov 2018).
URL `https://www.mountaingoatsoftware.com/agile/user-stories`

[14] J. Benesty, J. Chen, Y. Huang, I. Cohen, Pearson correlation coefficient, in: Noise reduction in speech processing, Springer, 2009, pp. 1–4.

[15] D. Phan, D. Vogel, J. Nunamaker, The search for perfect project management, Computerworld 22 (39) (1988) 95–100.

[16] M. Van Genuchten, Why is software late? an empirical study of reasons for delay in software development, IEEE Transactions on Software Engineering 17 (6) (1991) 582–590.

[17] A. L. Lederer, J. Prasad, Causes of inaccurate software development cost estimates, Journal of systems and software 31 (2) (1995) 125–134.

[18] G. H. Subramanian, S. Breslawski, An empirical analysis of software effort estimate alterations, Journal of Systems and Software 31 (2) (1995) 135–141.

[19] A. L. Lederer, J. Prasad, Information systems software cost estimating: a current assessment, Journal of information technology 8 (1) (1993) 22–33.

[20] A. R. Gray, S. G. MacDonell, M. J. Shepperd, Factors systematically associated with errors in subjective estimates of software development effort: the stability of expert judgment, in: Sixth International Software Metrics Symposium, 1999. Proceedings., IEEE, 1999, pp. 216–227.

[21] J. Cohen, Statistical Power Analysis for the Behavioral Sciences, Routledge, 1988.