www.ceejme.eu
ww.wsb.pl/wroclaw/ceejme

ISSN electronic version 2353 - 9119
Central and Eastern European Journal of Management and Economics
Vol. 4, No.4, 261-280, Dec. 2016

CEEJME
Central and Eastern European Journal of Management and Economics

Factors influencing user story estimations: an industrial interview and a conceptual model

Marek MAJCHRZAK, Lech MADEYSKI

Wroclaw University of Science and Technology, Poland

Abstract: A proper estimation of time in user stories is a crucial task for both the IT team as well as for the customer, especially in agile projects. Although agile practices offer a lot of flexibility and promote a culture of continuous change, there are always clearly de need timeboxed periods where an IT company has to commit to delivering working soft-ware. Estimating time of user story implementation provides clarity and the opportunity to control the project by the management, yet at the same time, it can increase pressure on software developers. Thus, incorrectly estimated user stories may lead to quality problems including system malfunction, technical debt, and general user experience issues. The paper describes user story characteristics, reasons of user story estimation inaccuracy as well as a model of their potential impact on post-release defects in large IT software ventures, all derived from the conducted interview with practitioners in Capgemini software development company.

Keywords: estimation, user story, agile, defect prediction, Scrum
JEL: C92, G31, D81, D83, C53

1. Introduction

In the 1995 Standish Group conducted research among IT companies (The Standish Group, 1995), they found that only 16% of the projects were completed on-time and on-budget. The last CHAOS report (The Standish Group 2015) showed the increasing rate, near to 40%, of agile projects successfully resolved. Despite the fact that the Scrum framework, at the team level, dominates (VersionOne 2015) among other agile methods and practices, one can observe growing importance of Development & Operations (DevOps) culture. The main reason for applying DevOps in the development process is the ability to release very often (i.e., daily or even several

Correspondence Address: Lech Madeyski, Wroclaw University of Science and Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland. Email: lech.madeyski@pwr.edu.pl; Marek Majchrzak, email: marek.majchrzak@pwr.edu.pl

times per day) (Puppet Labs and IT Revolutionary Press 2015) fully tested working software. To achieve that, especially in large IT ventures, one needs to introduce sophisticated testing as well as predictive analytics techniques (Buenen, Walgude 2016). Predictive analytics can be used to predict defect-prone software modules (e.g., classes). The aim of this paper is lay down the foundation for the novel kind of defect predictors, based on inaccurate estimations, that could be used to enrich software defect prediction models. Understanding the reasons of inaccurate estimation and its potential impact on software quality would also be beneficial for the practitioners.

Estimating the cost of a software development project, since the very beginning of software engineering (Caminer 1958, Brooks, Frederick 1987), is one of the most crucial tasks for IT companies. Unfortunately, it is still one of the project management aspects that has to be improved.

The number of studies has been conducted to understand inaccurate estimation phenomena in agile projects and possible effects on project schedule and budget. For example, Lang et al. (2011) looked at how classical problems that affect cost estimation in traditional software development projects are managed within the agile paradigm. They analysed several agile projects, and they found that estimation inaccuracy was a less frequent occurrence for these companies than in traditionally software development projects. They analysed several factors which may lead to inaccurate estimates. They found that potentially a severe threat to accurate estimates are user communication difficulties. Additionally, they notice that new people and new technologies in the team are a principal threat to produce accurate estimates. Many studies focus on project cost estimation inaccuracy. Liskin et al. (2014) analysed the single User Story (US) aspect and in particular the granularity (size). They found that the granularity of the US was seen as one of the main reasons for inaccurate estimation. Børte et al. (2012) analysed the role of social interaction in software e ort estimation. They proposed to consider the socio-cultural perspective in e ort estimation. Especially the ways in which software professionals reach a decision. They found, that during the planning meeting, US mediates between the historical practices and the use of generic and specific knowledge.

Knowledge of time spent on a task allows scheduling work which can be carried out in the nearest future. It is important for project managers who can control human resources in an appropriate manner to achieve the best possible results. On the other hand, historical data from past sprints may be used by an agile team as a hint during the planning meeting to avoid incorrect user

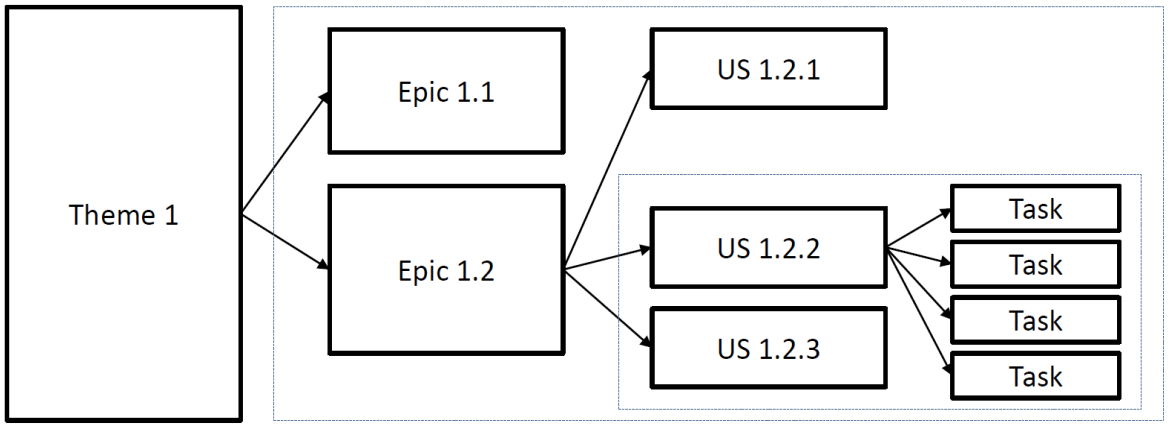
story (US) estimates and to mitigate the risks of sprint failure. Working under time pressure in every software project may lead to different quality problems. However, there are no studies that have analysed the impact of incorrectly estimated US on post-release defects in software. Hence, the aim of the paper is to fill this gap and to investigate the reasons of inaccurate estimate and impact on post-release defects.

This paper is organised as follows: a general description of the US life-cycle and characteristics in the Section 2. Section 3 describes the conducted survey, participants and presents the results, while the discussion is in Section 4. Conclusions and future work are presented in Section 5.

2. User story characteristics

In software development and especially in agile software development, a US is a high-level definition of a requirement (Ambler 2002) in the business language of the end user. User stories are a part of an agile approach which helps to shift the focus from writing about requirements to talking about them (Cohn 2004). From the business perspective, each US has a title, a description, an author, a group of stakeholders and its priority in the product backlog (PB). From the perspective of a development team, the characteristics include the size (estimate), tasks, acceptance criteria and a test plan. Each US has conversations that happen during backlog grooming and iteration planning to solidify the details. Additionally, especially in large enterprise ventures, US are grouped in the epics and themes (Cohn 2004) (see Figure 1).

Figure 1. Themes, epics and user stories



Source: based on Cohn (2004)

Refinement Process and Development. In enterprise projects, a US is prepared and then implemented in a long process supported by different experts and groups of stakeholders. Some of the phases are listed below:

- drafting the US,
- establishing agreements with key users - sponsor seeking,
- a prioritisation meeting,
- a refinement meeting,
- a rough estimate meeting, an estimation meeting,
- a planning meeting.

All of the parties concerned (stakeholders and the development team) are actively involved in the processes of both developments as well as testing. The testing process can be further distilled into:

- jUnit Tests (Development Team),
- end-to-end (e2e) tests (Development Team),
- integration tests (Development and Product Owner (PO) Team), acceptance testing (PO Team),
- user tests (Key Users and PO Team).

Such a level of arrangements (as the one described above) is required in the case of large IT systems on the basis of their dependency on various components as well as owing to the nature of contracts and arrangements between the client and the contractor.

In the ideal case, Scrum should be executed in the form of 'time and material', in reality, however, projects managed by an external company employ hybrid solutions (Zijdemans, Stettina 2014). Hence, for the both parties, a sprint becomes a fixed price contract *sui generis*. Thus, a US, which has been once agreed upon during the planning meeting, cannot be amended and has to be executed within the time frame predicted initially. If the budget is overrun, the costs are covered by the contractor.

The above process may, in fact, suggest that the contractor can enjoy a certain level of freedom in terms of estimating costs. This, in turn, may imply that they can assess risk as high in their estimations. However, this view does not hold entirely true. The client faced with strikingly high estimation prices may abort cooperation or renegotiate the terms and conditions of the contract, hence changing the cost of a man-hour or the cost of a story point.

*The problems with estimating user stories***.** A US receives the estimates from the development team before the sprint during the planning meeting. However due to the technical or business reasons, the estimate may by inaccurate. Thus, we can define the following types of estimation outcomes:

1. correct estimate (CE) - when the estimates are accurate or with a small margin of inaccurate estimation.
2. under estimate (UE) - when US requires more development efforts (over-runs) than set out in the estimate.
3. over estimate (OE) - when US requires less development efforts (under-runs) than set out in the estimate.

3. Survey and participants description

To find reasons of why the US are misestimated we conducted a survey among experienced IT professionals. The survey was conducted face to face as an individually recorded interview in the form of the discussion with authors of the paper. After that, collected insights and results have been analysed and categorised. We asked each participant two questions:

Q1. Why user stories are poorly estimated? Every suggested cause of misestimation was further investigated by posing question Q2.

Q2. What is the impact of the suggested cause of misestimation on the post-release defects?

Each participant gave several answers to Q1. After each answer, we immediately asked her or him about Q2. Despite the fact, the survey probe was relatively small (12 people) the results may be considered as valid because of significant and proofed record of people who have been interviewed (see

Table *1*). All of the survey participants have long experience both in agile (Scrum framework or Kanban technique) and waterfall IT projects.

Table 1. Survey participants

No.	Role	Experience
P1	SM, BA, SSD	8+ years of professional experience, 5+ years agile projects, team leader, automotive domain expert
P2	SA, QA Expert	10+ years of professional experience, 10+ years agile projects, leading architect
P3	SA, DBA	8+ years of professional experience, 5+ years agile projects, database expert, lead developer
P4	SSD	3+ years agile and professional experience, lead developer
P5	BA, SSD	7+ years of professional experience, 4+ years agile projects, automotive domain expert
P6	SM, PM	15+ years of professional experience, 5+ years agile projects
P7	BA, ST	8+ years of professional experience, 5+ years agile projects, automotive domain expert
P8	SM, PM, SSD	7+ years of professional experience, 5+ years agile projects, support team lead
P9	SSD, DBA	4+ years of professional experience, 4+ years agile projects, database expert
P10	PM	10+ years of professional experience, 3+ years agile projects
P11	BA	15+ years of professional experience, 3+ years agile projects, leading business architect
P12	PO, EA	15+ years of professional experience, 3+ years agile projects, leading product owner

Source : authors' own elaboration

Our aim was to cover estimation aspects from different perspectives. We selected different participants of the agile processes. We tried to get insights from both management perspective, including Scrum Masters (SM), Project Managers (PM), Product Owners (PO), domain specialists including Business Architects (BA), and strict technical staff, including Senior Software Developers (SSD), System Architects (SA), Enterprise Architects (EA), Database Administrators

(DBA) and Software Testers (ST). Another important aspect about the participants is that some of them have more than one role in the project or in the organisation. For instance, in a mature agile project, the role of the SM may be only part time. The SM may be responsible for other tasks like testing or business analysis. Most of the participants have been involved in the complex agile settings, in the automotive industry, like distributed and scaled Scrum or in Kanban-driven processes as described in (Majchrzak et al. 2014, Majchrzak, Stilger 2014).

4. Results

Answering Q1 was relatively easy for most of the participants. On the other hand, Q2 was much more difficult mainly because the projects does not match defects and the corresponding US in a regular manner, e.g., using JIRA issue linking[1]. Moreover, defects often appear as a result of the several US and several weeks after US implementation (long release cycles | even in agile projects), thus, the connection between a US (specifically between their metrics) and defects is not seen as obvious for agile project members. The answers including the reasons of poor estimations and have been grouped into 14 categories:

R01. Incomplete US life cycle.

R02. Level of knowledge and the authority of PO.

R03. Integration and integration testing with external systems.

R04. Technical problems and technical debt.

R05. Lack of domain knowledge.

R06. Developers' assignment change, sprint interruption and task switching.

R07. New developers in the team.

R08. Pressures from the management team.

R09. Granularity of the tasks.

R10. Level of US details.

R11. Incorrect organisation of PB.

R12. Organisational and coordination problems.

[1] Issue linking allows building an association between JIRA issues (US ← bug, task → task, bugs ← task, etc.). For instance, an issue may duplicate another, or the bug is related to given US.

R13. Estimates with/without safety buffer.

R14. Chain reaction.

The findings of particular IT experts are listed in Table 2. Based on Question Q2, in Table *3*, we have shown the possibility of post-release defects for the given type of estimation outcome (UE, OE) and the results category (R01…R14).

Table 2. Interview results

C/P	P01	P02	P03	P04	P05	P06	P07	P08	P09	P10	P11	P12
R01		x		x	x	x	x			x	x	
R02	x	x			x	x		x				
R03	x						x	x	x			x
R04		x	x	x	x				x	x		x
R05	x	x	x		x	x	x	x	x			
R06		x		x				x	x		x	
R07	x		x	x	x	x	x	x			x	
R08		x			x			x				
R09		x									x	
R10										x		
R11										x		
R12											x	x
R13		x									x	x
R14		x			x					x		

Source: authors' own elaboration

Table 3. Inaccurate estimation type and post-release defects

E/C	R01	R02	R03	R04	R05	R06	R07	R08	R09	R10	R11	R12	R13	R14
UE	x		x	x	x	x	x		x	x	x	x	x	x
OE				x			x							x

Source: authors' own elaboratiob

We described each category in details below. Each identified class consists of the obstacle description, reasons of inaccurate estimates and potential corresponding impact on post-release

defects. Often, depending on various conditions, both the US may be under or over estimate in each category.

R01. Incomplete US life cycle. As described in Section 2 each US has to go through the refinement process. The key team members (especially BA and lead developers) don't acquaint themselves with the specification. Often the US are complex and require a broad understanding of the underlying business domain, so that without support from the key team members, US may be incorrectly estimated during the planning meeting.Possible outcomes:*(UE)* - The PO and the team often don't see the whole picture. During the sprint planning meeting, the crucial business or technical details are forgotten. *(OE)* - Due to the late US acquittance the team becomes suspicious and uncertain. The fear of exceeding the time limit or missing the deadline often results in the estimation being higher. Often the high estimates do not reflect the reality. The development activities require less than expected.

R02. The level of knowledge and the authority of PO. Human factors always play a crucial role in the agile software projects. The team, in addition to the US description and acceptance criteria, needs to consider the past sprint work outcomes and problems during the sprint, both on the team and in the communication with PO and stakeholders. The crucial role has the PO. Part of the PO responsibilities is to have the vision of what he or she wishes to build, and convey that vision and domain knowledge to the team. It is the key to success in any agile software development project.

Possible outcomes:

(UE) - During the sprint planning meeting PO often gives the final US presentation. The absence of PO or lack of the detailed presentation induces that the team skips the key business requirements at the time of planning - extra time required for the clarifications.

(OE) - In a case of low PO authority, his or her absence or due to past experiences in terms of working with PO or key stakeholders, the team becomes un-certain. The team expects additional not specified workload, and thus, the tasks are being estimated with an extra safety margin. Often the high estimates do not reflect the reality.

R03. Integration and integration testing with external systems. Often the US requires integrating with other systems within given company landscape. Even though the modern technology used today (e.g. SOA - Service Oriented Architecture) offers well-defined interface contracts, integration is always time-consuming operation. Particularly in the large IT ecosystems

when the direct, peer-to-peer communication is not possible. Technical and security related details require a lot of e ort and are not visible in the US description.

Possible outcomes:

(UE) - occurs when it is not possible to effectively test interactions with a different system. The reason might be the lack of appropriate resources or a slow reaction to changes (a waterfall project plan on peer side). Due to inaccurate integration testing or even lack of such testing, what may happen are possible production problems.

(OE) - occurs when the team bases on their past experiences and therefore estimates every integration task with the high probability of failure. However often co-operation with another team is successful and result in the smooth execution of tests and arrangements to do with the interface contract. In such cases, an earlier completion of US does not affect the number of defects in the production.

R04. Technical problems and technical debt. The lack of knowledge in terms of the architectural limits, tools, and frameworks may lead to wrong assumptions and in turn results in a wrong estimation. Hidden technical debts that are often not recognised during the planning meeting imply a continued mistake making.

Possible outcomes:

(UE) - The knowledge de cit results in additional development tasks and in hidden defects which manifest themselves on the pre-production and production environments (e.g. in the case of significant load). Problems are not detected in the developers' environment. Technical debt results in defects are becoming more common. The time need for refactoring and additional features leads to a significant time overrun which in turn causes a lower involvement of such means as a review, integration and e2e testing.

(OE) - The lack of knowledge of the given framework or toolset often leads to pessimistic assumption that everything should be developed from scratch. Finding the right reusable solution results in significant underrun. On the other hand, improper use of the framework and omitting vital setting or activities may lead to underruns as well and in turn to post-release defects.

R05. Lack of domain knowledge. Understanding of a given field in agile software development is crucial from the user story development perspective. US detail is not set to 100% and requires the developer's know-how. Without understanding the domain, the development will not be efficient.

Possible outcome:

(UE) - The lack of understanding of a given domain leads to a significantly slower development. Also, the boundary conditions are often overlooked which leads to defects, especially in the production stage. Insufficient domain knowledge does not allow to run system and integration tests effectively.

(OE) - The potentially complicated matters of financing or legal requirements are often significantly easier in reality. Various concerns to do with knowledge deficits lead to an inflated time need during the estimation phase. However, more often than not, this does not affect future defects.

R06. Developers' assignment change, Sprint interruption, and task switching. Personal changes (in case of, e.g., someone's illness or unplanned tasks), in particular of a staff member responsible for a US, requires the onboarding of an in-coming employee.

Possible outcome:

(UE) - Switching or transferring tasks always affect productivity which in result may lead to an increased number of defects. A growing technical debt could spur a snowballing effect. In such cases, the deadlines are often missed, and this may have an impact on future defects.

R07. New developers in the team. In every enterprise software project, people rotate. Depending on the situation a new person may enter the project to increase the headcount, or one may be exchanged because of new project assignment. In both cases, new team member does not have the project-specific business and technical knowledge.

Possible outcome:

(UE) - Tasks are often planned by experienced programmers who do not take into account the fact that these may be given to new colleagues. The lack of technical and domain knowledge or expertise may extend the timeframe dedicated to a given task. Another issue may usually be posed by the lack of familiarity with the given system, the rules of testing, avoiding certain boundary conditions while testing or implementing, multiple code review cycles, or an excessive use of certain team members. Major consequence is a significant number of problems during the development and defects.

(OE) - New team members are often unplanned within the sprint or only part time. Occasionally, however, such team members may be very productive from the very beginning. It may be attributed to their participation in similar projects in the past. However, it is still feasible

that post-release defects will occur due to omitting crucial, project-specific boundary conditions in tests.

R08. Pressures from the management team. Occasionally, due to certain pressures applied by the management team, a task is underestimated despite the fact that it requires a substantial e ort, or when it is to be carried out by an inexperienced team. The management team accepts exceeding the time limit. Generally speaking, such a behaviour stems from the politics around a particular project, and the willingness to conceal certain issues by the contractor.

Possible outcome:

(UE) - Despite the green light from the management team, the awareness of exceeding time limits by developers in a given task impacts negatively on the given team causing stress and demotivation.

*R09. Granularity of the task*s. During the sprint planning meeting, the team turns the high-level US of the product backlog into the more detailed tasks. There are no strict rules how to de ne tasks, and thus, the team may freely determine the granularity of the work. For instance, if we consider the hypothetical US, where one has to implement the user CRUD[2] operations, we may get three different sets of tasks:

- *task 1: "User CRUDE"(12h);*
- *task 2: "User data access object"(8h) and "User hibernate mapping"(8h);*
- *task 3: "User data access object"(8h), "User hibernate mapping"(8h) and "User database (DDL[3] and DML[4])"(4h);*

If we now consider the estimated time, then we see the relatively big difference between the first and the third task set. Based on the third tasks set one will need 20 hours for the US realisation - 8 hours more than in the first set.

Possible outcome:

(UE) - US is turned into the relatively small number of tasks representing main US features. Sometimes without digging deeper into the details, some of the crucial work items are omitted. For instance, tuning the database would be probably forgotten during the sprint planning meeting in

[2] CRUD stands for Create Read Update Delete.
[3] DDL stands for Data Definition Language. It is used to create and modify the structure of database objects in a database.
[4] 4DML stands for Data Manipulation Language. . It is used to retrieve, store, modify, delete, insert and update data in a database.

the first case – but during the sprint, when it comes to the real implementation, the team will have to tune the database.

(OE) - Sometimes US has lots of small tasks for every single operation. The sum of the estimated times is often much higher than the sum of well split US.

R10. Level of US details. The definition of the US says that it should include a description of the business requirements, but more importantly, each US should include conversations (between users and the team) about the desired functionality. So, in other words, US should be simple and has to provoke the discussion. However some of the PO tends to provide a lot of detailed information almost on the source code level.

Possible outcome:

(UE) - Fewer details may lead to optimistic estimations. The team may not understand or remember all spoken details.

(OE) - Even though the detailed US is understandable for the team members, the number of details promote conservative estimations. Often such a US is easy to implement because of no need of extra clarifications and almost ready business solution.

R11. Incorrect organisation and structure of PB. As mentioned in the Section 2 the US are grouped in epics. In general, epics represent primary capabilities of the IT system. Later, at some point of project life-cycle, epics, step by step, are divided into the US. The USs from the same epic are often dependent on each other. Additionally, every item in the PB has priority. The priority indicates the implementation order.

Possible outcome:

(UE) - Both wrong implementation order and incorrectly divided epics lead to bottlenecks during the sprint, rework of the US and in turn unplanned tasks during the sprint.

R12. Organisational and coordination problems. If the project consists only of one team, then all disturbances and dependencies are managed effectively inside the Scrum team. In the case of large projects consisting of several Scrum teams, we have to take into consideration an entirely different network of connections and dependencies. The information exchange and agreements often are done late. The data flow to the particular Scrum teams may be ineffective.

Possible outcome:

(UE) - Missing link to the other Scrum team leads to software integration conflicts, ineffective testing, and rework of the implemented solution.

R13. Estimates with/without safety buffer. In general, every task estimate consists (in fact) of the following parts: the implementation, xUnit tests, manual tests, domain knowledge transfer and the safety buffer (risk). Depending on team constellation and external conditions the foreseen risk may be different.

Possible outcome:

(UE) - Technical leaders try to show their technical excellence to satisfy the customer and the management. They tend to force the removal of the risk aspect from the estimates. In the case of complex or new US (from technical or business perspective), it often leads to the significant overruns. High probability of overruns makes them introduce technical debt.

(OE) - Fear of failure, uncertainty or lack of trust inside the team may lead to conservative estimation with safety buffer. Instead following the lean thinking rules "think big, act small, fail fast; learn rapidly" (Poppendieck and Poppendieck, 2003), developers (especially with junior skills or the new people in the team) don't want to fail during the sprint. They want to finish the tasks in estimated time or even quicker. Usually, the foreseen risks do not occur. Thus, the US is finished faster than expected.

R14. Chain reaction. In the case of one or more USs-in-sprint being significantly extended, another USs may be realised in a way different than predicted during the planning stage. For instance, what might be lacking is the absence of key developers who need concentrating on the different US. Alternatively, what may hinder progress is enforcing of task completion in a shorter period, at the same time avoiding significant refactoring or creating a technical debt.

Possible outcome:

(UE) - The lack of key team members causes further extensions of the US. An inexperienced team often makes wrong decisions which in turn cause production defects.

(OE) - When working on a US, particularly when the sprint schedule is busy, certain activities or tasks are consciously skipped. These include proper testing, refactoring or code review. Hence, a task may take shorter than planned initially.

5. Discussion of the survey results

In this section, we describe the main groups of the inaccurate estimation reasons as well as the resulting post-release defect factors.

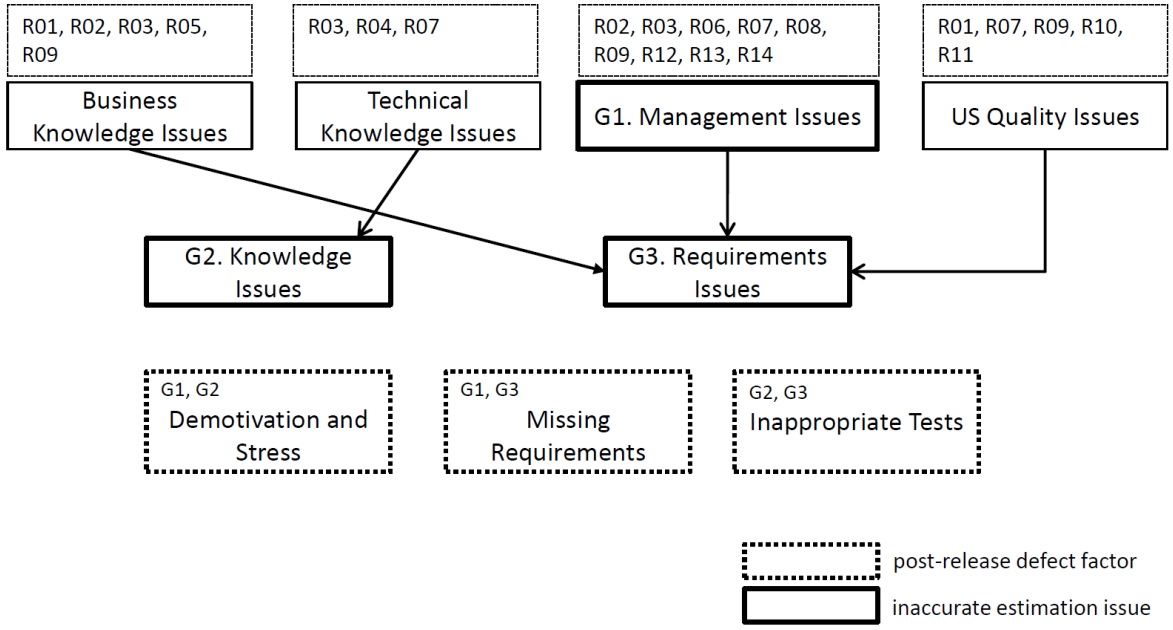
Reasons of inaccurate estimations

Similar to Lederer and Prasad (Lederer and Prasad, 1995) we grouped the findings into major categories. In their study, they identified four causes of inaccurate cost estimates in traditional (waterfall) projects: methodology, politics, user communication and management control. For both under and over estimated US we identified three categories (Figure 2 and Source: authors' own elaboration.

Figure *3*):

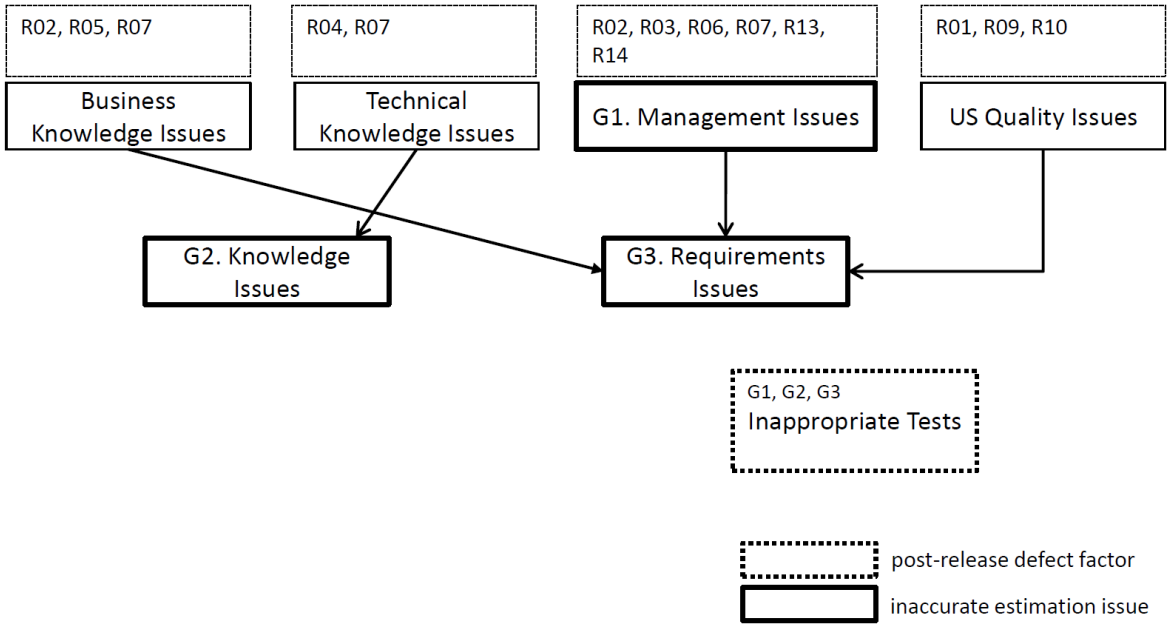
- *G1. Management Issues* - the project management (core) team (PM, SM, PO, BA, SA) doesn't plan the human resources with the proper safety buffer, does not prevent task switching and external disturbances. On the other hand, the development sub-teams may not share common planning strategy or don't use historical data.
- *G2. Knowledge Issues* - the team and more specific PO and Developers, do not have proper technical and domain background.
- *G3. Requirements Issues* - the US does not contain the mature description, acceptance criteria, test plan or does not have the right priority. Also in the case when the US was not included in the refinement process.

Figure 2. Under Estimation Issues and Post-Release Defect Factors



Source: authors' own elaboration.

Figure 3. Over Estimation Issues and Post-Release Defect Factors



Source: authors' own elaboration.

Our findings, in fact, are similar to Lederer and Prasad (Lederer and Prasad, 1995), but in contrary to them, we do not think that, on the level of sprint planning, the politic issues influence estimates of US. The reason behind it is following | developers who are responsible for estimating US focus rather on technical and domain related aspects rather than on organisational issues or political perspective of the venture.

Consequences of Inaccurate Estimations

Inaccurate estimate of US may results in post-release defects (

Table *3*). For each estimation outcome, base on interview results, we defined and described post-release defect factors. Post-release defect factors in case of underestimated US:

- *Demotivation and stress* - in general, defects often results from the programmer's fatigue. It often results in working days being longer than 8 hours or the need for working at weekends which increases the probability of system defects. Working under pressure, without keeping deadline de-motivates the team. Developers do not focus on continuous improvement aspects like continuous refactoring, continuous integration, and continuous feedback. As a result, the team creates incurring technical debt, which will manifest itself in the consecutive US. Developers do not fulfill Definition of Done and avoid the communication with external partners as well as inside the group.
- *Missing requirements* - working under pressure, from the both PO and the developments team perspective, increases the possibility of omitting requirement or boundary condition. If there is no time for proper discussion than the needs will not crystallise during the sprint. Missing requirements will be probably identified first on the production environment by end users.
- *Inappropriate tests* - approaching sprint end deadline results in inaccurate xUnit, end-2-end or integration tests or omission of such. The aim of agile software development techniques is the increased release frequency. Lack of appropriate regular testing, in short term period, will lead to post-release defects. xUnit tests, which have low mutation score (Madeyski 2007, Madeyski, Radyk 2010) or poor code coverage, will result in an inability to refactor the code and thus create incurring technical debt.

Post-release defect factor in case of overestimated US:

- *Inappropriate tests* - overestimated US are in most cases less dangerous, and, in general, do not cause post-release defects. It happens, however, that overestimation occurred as a result of skipping certain activities such as tests, design and code reviews, etc. Such an omission of vital activities (from the point of view of quality assurance and engineering techniques, particularly in agile ventures) may result in production defects.

Overestimated US, to some extent, are in most cases less dangerous, and, in general, do not cause post-release defects. Most of the survey participants saw the strong connection between underestimated US and post-release defects. On the other hand, one of the interviewed experts (P12), stated that even though the US may miss some crucial software engineering activities, in case the project has fully automated regression tests, most of the defects should be found immediately by the tests.

4. Conclusions and further studies

Most of the studies concentrate on the correctness of the project estimates. Looking at current software development trends, we observe that more and more projects are adopting the agile software development principles. The importance of the cost/effort estimates moves from the complete project perspective into the sprint level.

We conducted the interview with 12 IT professionals with proven agile expertise. We collected the reasons of inaccurate US estimation. We analysed their findings and tried to identify the estimation outcomes on the level of single US.

We identified, for both under and over estimated US, leading reasons of in-accurate estimation. In both cases we found three major groups: management issues, knowledge issues and requirements issues. Our surveys suggest that there are post-release defect factors that are directly related to inaccurate estimates. We find, that in the case of underestimated US, major role may play demotivation and stress, missing requirements and inappropriate tests. We also found that overestimated US in most cases will not cause post-release defects. Except if the team accidentally or due to lack of knowledge omits some crucial software engineering activities.

Knowing the potential connection between the inaccurate estimation and post-release defects, further research will be focused on the development of new process metrics, based on inaccurate estimation and useful in software defect prediction. We are currently conducting a large in-depth analysis of an industrial software project. Preliminary results suggest that new, estimation based metrics may improve software defect prediction models in agile settings.

Bibliography

Ambler S. (2002), Agile modeling: Effective practices for eXtreme programming and the unified process, John Wiley & Sons, Inc., New York.

Børte K., Ludvigsen S.R., Mørch A. I. (2012), The role of social interaction in software effort estimation: Unpacking the magic step between reasoning and decision-making. Information and Software Technology.

Brooks J., Frederick, P. (1987), No silver bullet essence and accidents of software engineering. Computer.

Buenen M., Walgude A. (2016). World quality report 2015-2016, 7th edition, Technical report, Sogeti and HP, Capgemini.

Caminer D.T. (1958), - and how to avoid them, "The Computer Journal", vol. 1 no. 1, pp. 11-14.

Cohn M. (2004), User stories applied: For agile software development, Addison Wesley Longman Publishing Co., Inc., Redwood City.

Lang M., Conboy K., Keaveney S. (2011), Cost estimation in agile software development projects, in: Information Systems development, reflections, challenges and new directions [Proceedings of ISD 2011, Heriot-Watt University, Edinburgh, Scotland, UK, August 24 - 26, 2011], pp. 689-706.

Lederer A.L., Prasad J. (1995), Perceptual congruence and systems development cost estimation, "Information Resource Management", vol. 8 no. 4, pp. 17-28.

Liskin O., Pham R., Kiesling S., Schneider K. (2014), Agile processes in software engineering and extreme programming: 15th International Conference, XP 2014, Rome, Italy, May 26-30, 2014. Proceedings, chapter: Why we need a granularity concept for user stories, pp.110-125.

Madeyski L. (2007), On the effects of pair programming on thoroughness and fault-finding effectiveness of unit tests, in: eds. Munch, J. Abrahamsson, P., eds, "Product-Focused Software Process Improvement", vol. 4589 of LNCS, pp. 207-221.

Madeyski L., Radyk N. (2010), Judy-a mutation testing tool for Java. Software, "IET", vol. 4 no. 1, pp. 32-42.

Majchrzak M., Stilger L. (2015), Experience report: Introducing Kanban into automotive software project, in: eds. Kosiuczenko P., Śmiałek M., From requirements to software: Research and practice, Scientific Papers of the Polish Information Processing Society Scientific Council, pp. 15-32.

Majchrzak M., Stilger L., Matczak M. (2014), Working with agile in a distributed environment, in: eds. Madeyski L. Ochodek M., Software engineering from research and practice perspective, Scientific Papers of the Polish Information Processing Society Scientific Council, pp. 41-54.

Poppendieck M., Poppendieck T. (2003), Lean software development: An agile toolkit, Addison-Wesley Longman Publishing Co., Inc., Boston.

Puppet Labs and IT Revolutionary Press (2015), 2015 State of DevOps Report, Technical report, Puppet Labs.

The Standish Group (1995), CHAOS Report 1995.

The Standish Group (2015), CHAOS Report 2015.

VersionOne (2015), The 10th annual state of agile report, Technical report, VersionOne Inc.

Zijdemans S.H., Stettina C.J. (2014), Agile processes in software engineering and extreme programming: 15th International Conference, XP 2014, Rome, Italy, May 26-30, 2014, Proceedings, chapter: Contracting in agile software projects: State of art and how to understand it, pp. 78-93.

Czynniki wpływające na estymacje historii użytkownika: wywiad w środowisku przemysłowym i model koncepcyjny

Streszczenie:

Prawidłowe szacowania czasu pracy dla *User Stories* jest istotnym zadaniem dla zespół IT jak również dla klienta zwłaszcza w projektach agile. Podejście zwinne oferuje dużą elastyczność i promuje kulturę ciągłych zmian, jednakże z punktu widzenia kontraktu zadania w pewnych okresach czasu musza być jednoznacznie wyestymowane. Szacowanie czasu realizacji *User Stories* zapewnia przejrzystość i możliwość kontroli projektu przez kierownictwo, jednak z drugiej strony, może zwiększyć presję na programistów. W związku z tym *User Stories,* które są niepoprawnie oszacowanie, mogą prowadzić do problemów związanych, z jakością oprogramowania, w tym awarii systemu i długu technicznego. Artykuł opisuje cechy *User Stories*, powody, dlaczego są błędnie oszacowane, jak również prezentuje model, który pokazuje potencjalnego wpływu na błędy powydaniowe w dużych przedsięwzięciach informatycznych. Dane pochodzą z dużych projektów realizowanych w metodykach zwinnych w firmie Capgemini.

Słowa kluczowe: estymacja, historie użytkownika, metodyki zwinne, predykcja defektów, Scrum

JEL: C92, G31, D81, D83, C53