Computer Science • 21(3) 2020 https://doi.org/10.7494/csci.2020.21.3.3618

Aleksander Jarzębowicz
Katarzyna Poniatowska

EXPLORING IMPACT OF REQUIREMENTS ENGINEERING ON OTHER IT PROJECT AREAS – CASE STUDY

Abstract *Requirements Engineering (RE) is recognized as one of the most important (yet difficult) areas of software engineering that has a significant impact on other areas of IT projects and their final outcomes. Empirical studies investigating this impact are hard to conduct, mainly due to the great effort required. It is thus difficult for both researchers and industry practitioners to make evidence-based evaluations about how decisions about RE practices translate into requirement quality and influence other project areas. We propose an idea of a lightweight approach utilizing widely-used tools to enable such an evaluation without extensive effort. This is illustrated with a pilot study where the data from six industrial projects from a single organization were analyzed and three metrics regarding the requirement quality, rework effort, and testing were used to demonstrate the impact of different RE techniques. We also discuss the factors that are important for enabling the broader adoption of the proposed approach.*

Keywords Requirements Engineering, Impact, Evaluation, Case study

Citation Computer Science 21(3) 2020: 261–286

1. Introduction

Requirements Engineering (RE) is one of the most important (yet difficult) areas of software engineering [4,6], as many RE-related problems and challenges are still being reported by practitioners [15,26]. It is also known to have a significant impact on other areas of IT projects and their final outcomes – numerous sources (e.g., [24, 26, 38]) describe the relationships between RE-related problems and many resulting project failures or challenges.

However, when it comes to empirical studies investigating RE's impact (i.e., the consequences of decisions about the RE process, applied techniques, practices followed, etc.), there are not many available studies. Naturally, there are many papers describing how a novel RE-related proposal (e.g., method, technique, or tool) is empirically validated through controlled experiments, action research, or case studies (e.g., [1, 19, 30]). The scope of such research studies is usually limited, however, as their purpose is to apply and evaluate the particular proposal under consideration. Also, a significant number of papers on comparing various RE techniques or practices with respect to the inherent characteristics (e.g., [22,41]), applicability in specific contexts (e.g., [5, 43]), or effectiveness in mitigating known problems (e.g., [18, 28]) can be found. Such works, however, usually focus on a relative comparison (sometimes empirically-based, sometimes more speculative) and rarely analyze the wider scope of the consequences implied by the use of a given practice. As a result, only a few sources that attempt to analyze a wider set of RE process variables and their outcomes in an industrial setting are available despite the fact that it is worthwhile to demonstrate RE's impact in a wider context [9].

The possible reasons for such a lack of empirical studies and quantitative data include the difficulty to measure or manipulate essential variables and the very high effort required [25]. Some examples that illustrate the scale of such effort include an 18-month project involving 9 companies [37], a 30-month project in a single company requiring a serious commitment of the researchers [7], or 2 large survey studies gathering data from more than 400 organizations [8].

Thus, it is difficult for researchers and even more for industry practitioners to make evidence-based evaluations that investigate how decisions about RE translate into requirement quality and influence other project areas. On the other hand, such evaluations can provide valuable information, both about the more generic picture of the effectiveness of RE practices among the IT industry as well as the opportunities of improvement for a particular company. By the latter, we mean focusing on the specific context and processes of a company and using the results of the evaluations to define the RE process for future projects. However, given the expected effort, it is rather unlikely that a company (at least a smaller one) that runs commercial projects would dedicate significant resources to such investigations, much less to conduct experiments related to their development processes [16, 20]. This situation could change, however, if the means for conducting such evaluations in a way that does not interfere with project tasks and does not require much effort were provided.

We tried to determine whether such means can be provided by utilizing the existing (and popular) approaches and tools. For this reason, we conducted a case study and analyzed data from six concluded projects from a single software company. From the data and additional background knowledge provided by the people involved in these projects, we were able to determine that the application of different RE practices translated to differences in the metrics reflecting the requirement quality, rework effort, and testing effectiveness.

Our work draws on the ideas of Agile experimentation [23], which proposes small-scale experimentation in real-life industrial projects. Such an approach is expected to provide more meaningful results than classroom experiments, for example, and minimize the risk that experimenting may impede the main goal (delivering software to the customer) or significantly increase the workload of project participants at the same time. The research described in this paper can barely be called experimentation – the pilot study we conducted was an ex-post analysis of the data from past projects. We can, however, see the opportunity to use the proposed approach in ongoing or planned projects for ex-ante predictions.

This paper is an extended version of a conference paper [14] that provided the following main contributions: (1) an industrial case study investigating the impact of RE practices on other project areas; and (2) a proposal of a lightweight approach to evaluate RE's impact in industrial projects, including experimentation in ongoing projects. The additional contributions of this extended version include the following: (1) inclusion of a larger set of related work sources; (2) a more detailed description of the conducted study, especially with respect to the evaluated requirements engineering techniques; (3) an additional section on validation activities involving RE experts; (4) an expanded section on threats to validity based on established guidelines; and (5) additional conclusions.

The paper is organized as follows. Section 2 provides more background by discussing the related work. In Section 3, an industrial case study based on six projects is presented along with its findings. In Section 4, we attempt to generalize the described case study into a universal approach that enables an evaluation of RE's impact and experimentation in an industrial setting. The paper is concluded in Section 5.

2. Related work

The work related to our research encompasses broader empirical studies focusing on RE and its impact. By broader, we mean those that take several variables into consideration (as opposed to studies dedicated to a particular technique, which is a common way of validating new proposals). Such empirical studies can be done in many ways, including (1) those more similar to our work (e.g., case studies, action research, or controlled experiments) and (2) questionnaire surveys or interviews. Below, we summarize the research belonging to each of these two groups.

Sommerville and Ransom [37] conducted a study during a project from the EU Framework Program. Its main purpose was to improve the RE processes of nine soft-

ware companies belonging to a project consortium. The metrics collected by them showed the improvement (of varying degrees) in all companies as well as better business performance. Damian and Chisan [7] focused on the RE processes of a single organization during a 30-month study. They analyzed the processes, suggested improvements to their practices, and evaluated the outcome, noticing a positive impact on other project areas as well as on the general productivity, quality, and risk management. Kamata and Tamai [17] investigated the correlation between the completeness and quality of software requirement specifications (SRS) and project success (in terms of schedule and budget). For this reason, they reviewed 32 SRS documents using the IEEE 830:1998 standard [10] as a reference point. The results showed that successful projects had a more balanced coverage of various requirement categories and that the purpose, product perspective, and functions sections of SRS were crucial. Radliński [31] applied data-mining techniques to a large set of IT project data (ISBSG dataset) to identify the correlations among RE practices (e.g., RE techniques, documentation developed in a project) and the overall project results. Rapp et al. [32] developed a questionnaire-based method of RE process assessment and validated it by assessing several industrial and semi-industrial projects. They also stressed that such a method must minimize the assessment effort and provide a real business value to the assessors and organizations undergoing the assessments. Bormane et al. [3] analyzed 12 software projects from a major Latvian company, comparing Waterfall/Agile and small/large projects. They found that the Agile approach required much lower cost of change and enabled a more accurate effort and budget estimation in the case of large projects. They also analyzed elicitation techniques used in Waterfall and Agile projects but were not able to draw conclusions from the datasets used. Liechti et al. [21] proposed an approach for a continuous process improvement for Agile teams that uses analytical dashboards based on data from software supporting tools, for example. Their paper, however, does not provide many details on the metrics used and impact observed; this is possibly due to non-disclosure agreements.

Verner et al. [39] surveyed practitioners from Australia and the U.S., gathering data from 164 projects. Their findings indicate that effective requirement elicitation and requirement management are the most important contributing factors to project success. Bjarnason et al. [2] performed an interview-based exploratory case study investigating gaps in the communication of requirements. Among other things, their study revealed that the consequences of such gaps affect the requirement quality, failure to meet stakeholder expectations, and testing efficiency (for example). Sethia and Pillai [35] used responses from an online survey to investigate the influence of the general categories of elicitation issues (problems) on project performance. They applied several statistical techniques to identify the most influential issues. Ellis and Berry [8] conducted two worldwide surveys investigating the correlation between the maturity of RE processes and project success (in terms of schedule, budget, and product scope). The first survey allowed them to build an RE maturity model, while the second demonstrated a positive correlation between RE maturity and the ratio of successfully completed strategic projects. Mossakowska and Jarzębowicz [27] con-

ducted a survey to identify which RE techniques are perceived by practitioners as contributing to particular quality characteristics of a final software product.

In general, there are not many sources that could be considered to be directly related work. Some of the studies on RE impact are based on surveys and interviews rather than on a quantitative analysis of ongoing or concluded software projects. Among the remaining ones, most studies required a large workload to conduct, which contradicts our postulate to minimize the effort. The studies most similar to ours are [3] and [17], which analyze the data from concluded projects and seek correlations between RE and the outcomes measured at the level of a whole project. However, we used different metrics; moreover, we outline a more general approach to be used in further studies.

3. Industrial case study

We intended to investigate the feasibility of using the existing data from popular software tools to evaluate the impact of RE practices. For this reason, we conducted a pilot study using the available data from a particular software company that we approached. This section provides the background information about the company and its projects, describes the design of our study, and reports the obtained results.

3.1. Background

The research was conducted in a medium Polish company employing around 200 people. The company specializes in the development of electronic measuring devices (dedicated to various utilities like water, heat, gas, etc.) and related software systems (web and mobile applications for the configuration and administration of devices, reading the measurements through a radio channel or other conduits, and processing the measurement data). The customers of such software are both internal and external; the external ones include domestic and foreign organizations.

The software development projects run by the company differ with respect to their size and the methodology used, which led us to narrowing our focus to more specific types of projects to be included in our study. The following description of the development processes is also narrowed down to these types of projects. Such projects are developed in small teams (six to nine people), including one or two analysts responsible for requirements. They follow a hybrid development approach based on an incremental software lifecycle model and integrate selected plan-driven and Agile practices. This approach was most common for this company's projects in recent years. It was adopted as result of an intent to introduce Agile practices into development processes based on plan-driven methods.

This hybrid development process can be summarized as follows. The analysis phase is, to a large extent, completed at the beginning of a project and results in capturing the project's scope and generic requirements. The software product is then divided into a number of smaller components (to be developed in separate increments).

Each increment starts with establishing and documenting the detailed requirements for the component under consideration. Such a document is called a detailed software requirement specification. It uses a standardized template based on IEEE 830:1998 [10] (despite the fact that this standard was superseded by ISO/IEC/IEEE 29148 [13]). The template was developed by the company for internal use in all of their projects. The requirements are then reviewed by other interested stakeholders for the purpose of verification (mostly by developers and testers) and validation (by customer representatives). Verification by team members is supposed to check the quality of the requirements (i.e., whether they are unambiguous, testable, etc.). After reviews and possible corrections, the requirements are passed on to the developers, who work on the code and (when in doubt) contact the analysts for clarification. When the component is ready, it is passed on to testers, who verify its quality. This is followed by a phase of fixing any defects and stabilizing it, resulting in concluding the work on the component. Another increment dedicated to another component and following the same workflow starts subsequently. Such a process continues until the whole product is ready; then, delivery to the customer takes place. The Agile practices are mostly related to the work of the development team and include self-organizing teams, a prioritized work list, daily stand-up meetings, and retrospectives.

The software projects run by the company are supported by integrated Confluence[1] and Jira[2] tools (both provided by Atlassian) as well as an associated time-tracking plugin. Confluence is a software tool based on the idea of a wiki, enabling collaborative work on shared documents. It is used by analysts to document all requirements (both generic and detailed). The requirements are available to other project participants (like developers or testers) who can ask questions or report requirement defects by adding comments to particular fragments of the specification. The company's policy is to register each query related to the requirements. In addition to the versioning mechanism provided by Confluence, this allows us to track the evolution of the requirement specification (including fixing the defects of particular requirements). All such changes are also timestamped. Jira is an issue-tracking tool that can be used for bug tracking and/or managing Agile development. In the software projects of the discussed company, Jira is applied for both purposes. Apart from the typical use of defining the issues representing the product features and development tasks (plus updating their statuses and reporting the work effort), requirements engineering activities are also strongly supported by this tool. The project manager defines an issue corresponding to the task of the software requirements specification development, and the analysts report the work effort dedicated to this task by registering the particular activities they conducted along with the time spent on each. Moreover, if any other project member participates in the work on requirements (e.g., by discussing them or providing some input), then he/she is obligated to report the time spent on it. Such information is registered and associated with this issue. When the requirement

[1] https://www.atlassian.com/software/confluence
[2] https://www.atlassian.com/software/jira

specification is ready to be verified, the analyst creates new issues dedicated to the reviews and assigns them to the developers, testers, and (if necessary) other team members so the status and time effort of each review are registered in Jira as well.

3.2. Selected projects

As previously mentioned, we selected a set of similar projects from the whole portfolio of software projects completed by the company. The reason was that we intended to focus on RE and investigate how the differences in RE practices impact other project areas. Analyzing software projects as case studies is not easy, as each project is like a complex living organism with many influencing factors, both internal and external. We wanted to avoid comparisons between significantly different projects (with respect to size or development methodology, for example), as such differences could possibly obscure the phenomenon we investigated.

We selected six software projects conducted during the period of 2014–2017. The selected projects are summarized in Table 1. Each of them took from 9 to 18 months to complete. They all followed a similar development process (described in Section 3.1). Other similarities between them included product type and size (measured by the number of requirements) and the experience of the analysts responsible for RE (junior analysts supervised by a senior analyst). As for the size, some differences can clearly be noticed, but these do not exceed 25% (and it is difficult to expect equal values from real-life examples).

More-significant differences concern the requirements (in particular, the techniques of requirement elicitation and requirement documentation). The "Req. elicitation" row of Table 1 enumerates the elicitation techniques used by the analysts in a given project. The following elicitation techniques were used (the descriptions were adopted from recent industrial standards on requirements engineering and business analysis):

- Interviews – An interview is a formal or informal approach to elicit information from stakeholders that is performed by a business analyst by asking prepared and/or spontaneous questions and documenting the responses [29]. This technique is very interactive and allows for modifying the order of the previously prepared questions depending on the interviewees' answers and the situation [33]. Interviews are often conducted on an individual basis between an interviewer and an interviewee but may involve multiple interviewers and/or interviewees [29]. An interview can also be used for establishing relationships and building trust between business analysts and stakeholders in order to increase stakeholder involvement or build support for a proposed solution [11].
- Prototyping – Prototyping is a technique that enables us to obtain early feedback on requirements by providing a working model of the expected product before building it [29]. Prototyping is used to identify both the missing or improperly specified requirements and unsubstantiated assumptions by demonstrating what the product looks like and how it acts in the early stages of design [11]. The

purpose of prototyping in requirements elicitation is exploring the requirements by encouraging stakeholders to consent or object or to clarify and amend [12].

- Observation – Observation is a technique that provides a direct way of viewing people in their environments to see how they perform their jobs or tasks and carry out processes [29]. This consists of watching the users' activities and processes and identifying the system requirements on this basis [33]. On-site observation is conducted by watching the users working and documenting the processes, tasks, and results; in some cases, however, observation is extended to interviewing users about their jobs and how they accomplish their tasks [33]. This technique is particularly helpful for detailed processes when stakeholders experience difficulties or are reluctant to articulate their requirements [29].
- Document analysis – Document analysis is used to elicit information (including contextual understanding and requirements) by examining the available materials that describe either the business environment or the existing organizational assets [11]. Examples of documents include process models and maps, process descriptions, organization charts, product specifications, work procedures, standards, and instructions, templates of documentation, etc. [33].
- Workshop – Workshops bring stakeholders together in order to collaborate on achieving a predefined goal [11]. Workshops involve people who have different points of view on a given problem and help determine and describe the requirements coming from various perspectives [33]. This is a primary technique for quickly defining cross-functional requirements and reconciling stakeholder differences [29]. Moreover, workshops can be used to establish a solution's scope: to discover any hidden requirements, to define the priorities of the requirements, or reach a consensus when it comes to reaching an agreement on the requirements [33].
- Interface analysis – Interface analysis is used to identify where, what, why, when, how, and for whom information is exchanged between solution components or across solution boundaries [11]. The interface can be understood as a user interface of the developed solution but also as an API/data interface between two systems or an interface between business processes [11]. An analysis of user interfaces can utilize more-specific techniques like wireframing or display-action-response models [29].
- System archaeology – In system archaeology, requirements are extracted from existing systems: legacy systems as well as competitor systems or even analogous systems (systems in a different context with a similar functionality). This technique is mainly used if an existing system has been used (and possibly changed) over many years and is now to be replaced by a new system. This is especially useful if no other current documentation is available for discovering what a system really does [12].

One additional explanation is necessary with respect to the document-analysis technique applied in some of the considered projects: SIWZ is the Polish abbrevia-

tion of a document type used in public projects. The public organization acting as a project's customer is obligated by Polish law to publish the request for proposals and the terms of reference. The latter are described in a SIWZ document (and its attachments) and can include quite-detailed information about the requirements.

Table 1
Summary of selected projects

Project	P1	P2	P3	P4	P5	P6
Team size	6	9	7	7	7	10
No. of business stakeholders	3	4	1	1	2	1
No. of requirements	41	48	44	47	39	52
Req. elicitation	interviews, prototyping	interviews, observation, prototyping	document analysis (SIWZ), interviews	workshop, observation	document analysis (SIWZ), interface analysis	document analysis (SIWZ), system archaeology
Req. documentation	user stories, tables	tables, UC	tables	tables, UC	tables	tables
No. of req. defects reported	11	12	16	13	14	21

As for requirements documentation ("Req. documentation" row of Table 1), the following techniques were used:

- Table – a tabular requirement template with the following fields: ID, type (functionality, security, constraint, etc.), source, and description. The description was expressed in a natural language without any internal structure.
- UC – use cases providing a structured description of the functional requirements with detailed interaction scenarios (basic and alternative).
- User stories – a simplified form of expressing requirements: as a (role), I want (capability) so that (goal).

All of the analyzed projects (P1-P6) used tables to represent the requirements; however, some of them refined the functional requirements to the use cases (P2, P4), and in one project (P1), user stories were additionally used because the project team wanted to try out a new approach for documenting the requirements.

3.3. Study design

The data from Jira and Confluence can be exported to a spreadsheet format. We received the data from the above-mentioned projects in such a form and used it to conduct our study. Additionally, we could contact some people involved in these projects who were able to provide us with background information that is not necessarily registered in the supporting tools (e.g., the requirement elicitation techniques applied or stakeholder attitude).

We started this research with the intent of exploring the quantitative impact of RE on requirement quality and on the work done in other areas of an IT project. We had not formulated more-specific research questions a priori, because such a study was dependent on the data we could obtain (unlike with a controlled experiment). For instance, if data about requirement-related defects was not available (because a company's procedures had not required reporting them or the supporting tool does not allow us to distinguish them from other issues), it would be impossible to investigate any consequences of such defects. The scope of the reliable information we could gather from people involved in these projects was limited – for example, an analyst is likely to remember the elicitation techniques he/she applied in a given project but not the precise amounts of time spent on particular tasks. Thus, we would not be able to use this metric in case the time was not registered by the tools.

After the initial examination of the data and gathering background information related to the investigated projects, we were able to define the following three research questions:

- **RQ1**: Does the approach to requirement elicitation affect the quality of the requirements?
- **RQ2**: What is the impact of requirement quality on rework that could have been saved?
- **RQ3**: Does the requirement documentation technique affect the effectiveness of the testing?

Each research question was refined for the purpose of operationalization. We used the GQM (Goal-Question-Metric) method [36] to identify the measurements to be made.

3.3.1. RQ1

The requirement elicitation approach serves as an independent variable here. As shown in Table 1, we were able to identify the particular elicitation techniques used in each project. However, a more fundamental related issue was communicated by company employees. In some of the investigated projects (P3, P5, P6), the main source of requirements was a document prepared by the customer (SIWZ), and the communication with the customer representatives was very limited (by phone only; moreover, stakeholders were often reluctant to cooperate and required a significant communication effort from the analyst). In these projects, the most common technique applied was

document analysis. Additionally, analysts used other techniques like interface analysis or system archaeology (on competitor systems) that could provide them with information that was difficult to get directly from the customer representatives. In the remaining projects (P1, P2, P4), the analysts were able to act more actively and meet customer representatives, organize requirement elicitation sessions, and get to know the environment in which the prospective users worked. The stakeholders were rather cooperative; techniques like interviews, workshops, prototyping, and observation were used. We decided to distinguish two general approaches to requirement elicitation; namely, active (P1, P2, P4) and passive (P3, P5, P6). We were also able to investigate whether any difference between the quality of the requirements elicited by them could be noticed.

To measure the requirement quality, the following GQM structure was used:

- **Goal**: Assess the quality of the requirements.
- **Question**: How many defects does the requirement specification contain?
- **Metric**: Percentage of requirements for which defects were reported by the project participants.

We verified that the data exported from the tools allowed us to trace the comments containing defect suggestions, doubts, questions, etc. associated with particular requirements as well as the history of the changes made to the requirements. Additionally, we decided to categorize the defects related to the requirements. This task had to be done manually, however, because no such categorization was used when reporting defects in the considered projects. As a result, this turned out to be the most effort-consuming task of our study. We defined the defect categories as the violations of the particular characteristics of requirement quality. The following set of requirement quality characteristics, adopted from [13] (Section 5.2.5) and [11] (Section 7.2.4), was used:

- **Consistency**: The requirement is free of conflicts with other requirements.
- **Unambiguity**: The requirement is stated in such a way so that it can be interpreted in only one way.
- **Atomicity**: The requirement is self-contained and capable of being understood independently of other requirements or designs.
- **Correctness**: The requirement addresses an actual stakeholder's need and defines an essential feature of the system or a constraint.
- **Testability**: The requirement has the means to prove that the system satisfies the specified requirement.
- **Feasibility**: The requirement is technically achievable and fits within system constraints (e.g., cost, schedule, technical, legal, regulatory) with acceptable risk.
- **Completeness**: The requirement has the appropriate level of detail to guide further work and needs no further amplification because it sufficiently describes the stakeholder's need.

3.3.2. RQ2

The second question explores the correlation between requirement quality and rework. Requirements quality is measured in the same way as in RQ1 – by the percentage of requirements that contained defects. The rework, understood as the work that has to be done because of any defects (and could possibly be saved in the absence of such defects), turned out to be difficult to measure. When analyzing the data about the effort reported by the project participants, it was impossible to reliably distinguish the ordinary work and rework caused by defects. We were only able to capture it in a partial manner by identifying (on the basis of background information provided by the project participants) the people who were not initially involved in the considered projects but were engaged later on a temporary basis to deal with the consequences of the requirement defects. This group included domain experts, additional analysts, and developers, for example. We are aware that this operationalization does not fully reflect the phenomenon we intended to investigate; still, we believe even such a partial analysis can be interesting. Consequently, we derived the following GQM structure:

- **Goal**: Assess the amount of rework caused by requirement defects.
- **Question**: What was the proportion of additional person-hours?
- **Metric**: The number of person-hours reported by additional participants/the total number of person-hours initially planned for RE activities.

3.3.3. RQ3

In this research question, we intended to find out whether different requirement documentation techniques affected the work of the testers. As shown in Table 1, three techniques were used in P1-P6: tables, use cases, and user stories. Among these three techniques, use cases are much more detailed representations than the remaining ones. We decided to compare projects where use cases were applied with those remaining. The rationale was that more-detailed requirement specifications can help testers design more-thorough test cases and test scenarios (both positive and negative). To quantify the impact on the work of the testers, we proposed the metric of testing effectiveness, which is defined as the proportion of software bugs found during the internal testing by the company to all of the software bugs uncovered in the product (those found during internal tests and those reported by customers or other external parties). As the products of all of the considered projects were delivered to customers, stabilized, and used for some period of time, such external bug reports were available. We counted all of the issues assigned to the bug report category and distinguished them according to the source (internal/external). The GQM structure for this part of our research was as follows:

- **Goal**: Assess the impact of the requirement documentation technique on the effectiveness of testing.
- **Question**: What is the difference in the effectiveness of testing between the projects that applied use cases and the remaining projects?

- **Metric**: The proportion of bugs found during the internal testing to all bugs found in a given software product.

3.4. Study results

The investigation of defects in requirements (RQ1) allowed us to identify the erroneous requirements and categorize them. Most of the defects were reported before implementing the requirements (e.g., developers asking for explanations of ambiguous requirements and testers questioning requirement testability), but there was a minority of defects registered after implementation (e.g., feedback from a business stakeholder about a wrong software feature that was traced back to an incorrect requirement). The summary of our findings is shown in Table 2. An additional visualization is provided in Fig. 1, which depicts the percentage of erroneous requirements and distinguishes the projects according to their general approaches to requirement elicitation (active/passive). An observation can be made that those projects where an active approach to requirement elicitation was used had relatively fewer defects.

Table 2
Requirement defects in analyzed projects

Project	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**
Elicitation approach	active	active	passive	active	passive	passive
Total no. of reqs	41	48	44	47	39	52
No. of erroneous reqs	11	12	16	13	14	21

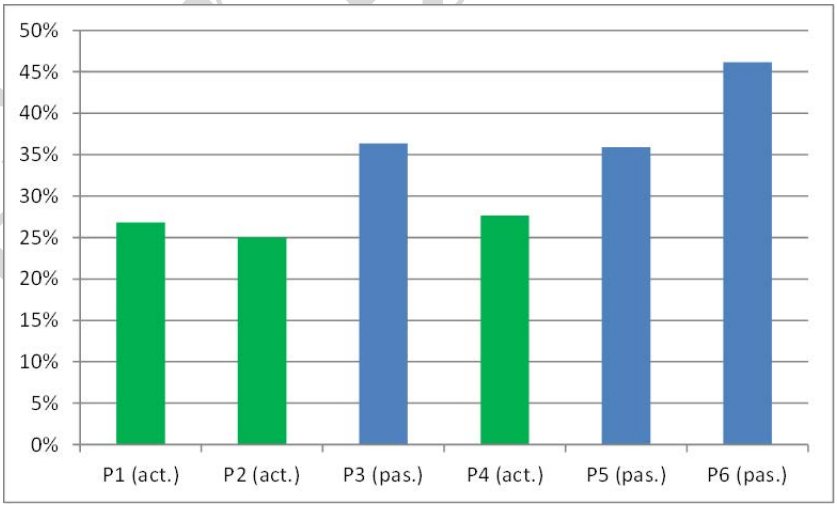

Figure 1. Percentage of requirements containing defects in analyzed projects

We conducted a further investigation to try to determine which kinds of requirement defects were encountered in particular projects. Each defective requirement was reviewed and assigned a defect type that was consistent with the categorization introduced in Subsection 3.3.1. The results are presented in Table 3 and additionally visualized in Fig. 2.

Table 3
Categorization of defects found in analyzed projects

Category:	P1	P2	P3	P4	P5	P6	Total
Consistency	2	3	3	–	1	2	10
Unambiguity	4	–	5	3	1	6	19
Atomicity	–	2	1	5	3	3	14
Correctness	3	2	3	1	1	4	14
Testability	–	2	1	4	3	3	13
Feasibility	2	–	2	–	3	1	8
Completeness	–	3	1	–	2	5	11
All	11	12	16	13	14	21	

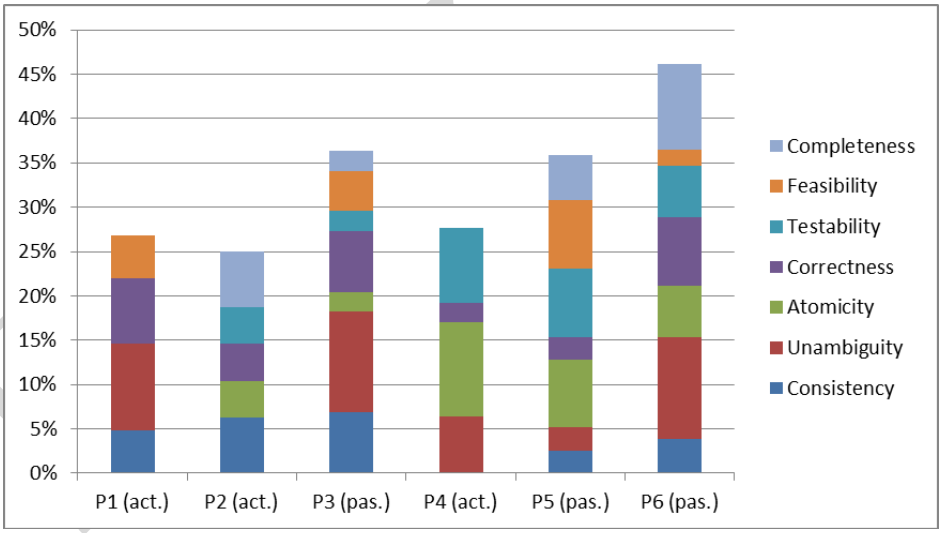

Figure 2. Percentage of requirements containing defects with defect classification

The most frequent kind of defect concerns unambiguity (in both active and passive projects; however, the former are not as affected as the latter). One possible explanation is that an active approach forces an analyst to ensure that the requirements that he/she elicit from a stakeholder are mutually understood in the same way.

On the other hand, an active approach results in more inconsistencies. This is probably due to the fact that requirements are elicited from many stakeholders representing different viewpoints while, in a passive approach, the main source of requirements was a document that could have its deficiencies but at least was internally consistent.

The most surprising result was that an active approach resulted in fewer defects concerning feasibility. We discussed this issue with a group of analysts (with 11, 7, and 5 years of experience, respectively). The discussion led to the conclusion that the document analysis elicitation technique can possibly decrease the awareness of an analyst – he/she can make an unfounded assumption regarding feasibility (that, if a requirement was included in an official document, it can be implemented) and refrain from consulting it with developers.

An exploration of RQ2 led to a juxtaposition of the requirement quality and RE rework effort, which is presented in Table 4. To represent quality, the percentage of erroneous requirements was used; this was computed on the basis of the values from Table 2. The rework is represented by the number of person-hours reported by additional personnel who had to contribute to the project. As already mentioned, we are aware that this representation is imperfect, but we were not able to extract more-accurate measurements from the available data. In Table 4, we also show the relative value of RE rework by comparing the effort reported by additional personnel to the total effort for the RE activities included in a project plan.

Table 4
Additional RE effort in projects

	Erroneous reqs	**Additional personnel**	**E1: Effort of additional personnel (person-hours)**	**E2: Effort planned for RE (person-hours)**	**E1/E2** (%)
P1	26.8%	Domain expert: 1; Developer: 1; Analyst: 1	7	168	4.2%
P2	25.0%	Domain expert: 1	5	176	2.8%
P3	36.4%	Domain expert: 1; Developer: 1; Analyst: 1	19	168	11.3%
P4	27.7%	Domain expert: 1; Developer: 1; Analyst: 1	3	182	1.6%
P5	35.9%	Domain expert: 1; Developer: 1; Analyst: 2	16	176	9.1%
P6	40.4%	Domain expert: 2; Developer: 1; Analyst: 2	31	104	29.8%

It can be noticed that the more erroneous requirement specifications resulted in more work for the additional people (especially in the case of P6, where the overhead reached nearly 30%); however, this is not a simply proportional dependency.

As indicated in the description of RQ3 in Section 3.3, we compared the effectiveness of the testing between projects that had functional requirements represented as use cases with the remaining projects. The results, which are summarized in Table 5, show the difference – projects with use case specifications had more than 80% of the bugs found by the company's testers, while for the other projects, the effectiveness of the testing was about 70%.

Table 5
Testing effectiveness in analyzed projects

Project	P1	P2	P3	P4	P5	P6
Use cases	no	yes	no	yes	no	no
Testing effectiveness	75%	87%	71%	84%	67%	69%

A further investigation via conducting interviews with three testers (who participated in some of the analyzed projects) revealed that they considered use cases as the most helpful requirement documentation technique and criticized other techniques, especially user stories (which they found too vague without well-defined acceptance criteria). Use cases also allowed testers to save much work by reducing the need to contact the analysts (and request clarifications) and by the fact that use cases can relatively easily be transformed into test cases. This finding is, however, based on interviews only, not the quantitative data derived from tools. In addition, we include some direct quotes that express the viewpoints of the testers we interviewed:

- *I appreciate when I get a specification of a use case including the main flow and alternate flows. Usually, it is enough to slightly rewrite them to develop corresponding test cases.*
- *The greatest problems I have are caused by user stories because analysts in our company don't have the habit of defining acceptance criteria. I find use case specifications very useful; they allow me to perform functional tests in a very effective and time-efficient manner.*
- *The best situation possible is when an analyst provides a requirement with traces to associated use cases. The use cases should be described using main and alternative scenarios; this makes it easier for me to understand a system's functionality. User stories did not work out for us, but this could be due to the fact that the team was not prepared for Agile development at the time.*

3.5. Validation

Validation is a term that is frequently used in scientific papers; however, it is usually in the context of demonstrating that a given proposed solution is effective or has some other expected properties [42]. In our case, the purpose of validation was different, as

we intended to find out whether our study results can be considered valuable from the point of view of RE practitioners and consistent with their experiences. We selected interview-based expert opinions as a method of validation and conducted interviews with two experienced RE practitioners (hereafter referred to as Practitioner 1 and Practitioner 2). Practitioner 1 had nine years of industrial experience with diverse responsibilities (two years in software testing, five in RE, and two in project management), mainly in projects delivering IT solutions for large corporations. Practitioner 2 had ten years of industrial experience as an IT analyst, mainly in software projects for customers from the financial domain. Before conducting the interviews, each interviewee received a written document with the description of our work (an extended version of the information provided in Sections 3.1–3.4). The interviews were conducted separately while using the same set of previously prepared questions. The main questions are preceded by introductory questions verifying whether the descriptions of the study and its results are clear and sufficiently comprehensive to the interviewees. The questions and answers obtained are summarized in Tables 6, 7, and 8 (each dedicated to a separate RQ).

Table 6

Expert opinions regarding RQ1 (impact of elicitation approach on requirement quality)

Practitioner 1	**Practitioner 2**
Q1.1: Is the purpose and design of the conducted research study comprehensible?	
Yes, I know such issues from my experience, I understand the dependencies described.	I had no problem with comprehending the study's aim and design.
Q1.2: Is the presentation of the results obtained in this study clear and unambiguous?	
Tables and graphs were most helpful to capture the dependencies found by the study.	Yes, the results can be easily read from the tables and figures. Their interpretation is straightforward as well.
Q1.3: Are the results obtained consistent with your professional experience?	
Yes, I noticed that, when I include workshops and interviews in the elicitation process, writing down requirement specifications is easier, which also translates into its better quality. And when developers have some questions or issues regarding these requirements, sorting them out takes me less time.	I agree that direct face-to-face contact with the customer is beneficial to the specification document. Personally, I prefer to specify requirements on the basis of my own notes taken during meetings with stakeholders, as it makes me aware of their intent. It gives me more confidence when specifying requirements and improves their quality.
Q1.4: Do you consider these results to be useful for IT projects practitioners?	
Yes, I believe it is worthwhile to show these results to project managers and customers to demonstrate which elicitation techniques can be used to achieve high-quality requirements.	Yes, I would gladly replicate such an analysis in my company. The results are interesting for the analysts and show them directions to improve their competencies.

Table 7
Expert opinions regarding RQ2 (impact of requirement quality on rework effort)

Practitioner 1	**Practitioner 2**
Q1.1: Is the purpose and design of the conducted research study comprehensible?	
To be honest, I had to read it twice; finally, I understood the meaning of rework consuming project reserves.	Yes, it was described clearly to me.
Q1.2: Is the presentation of the results obtained in this study clear and unambiguous?	
I know this topic from my experience, and I had no problems when reading and interpreting the results.	Yes, I understand the outcome of this research.
Q1.3: Are the results obtained consistent with your professional experience?	
I do not know how exactly requirement quality influences the rework effort, but, for sure, poorly elicited requirements cause delays. We have to spend additional hours of work on consulting domain experts or the customer.	Yes, well-specified requirements definitely minimize the time spent by programmers on adjusting system functionalities, for example.
Q1.4: Do you consider these results to be useful for IT project practitioners?	
Yes I do. Research like this can make managers realize that it pays to dedicate time to train analysts because the quality of the requirements impacts a project's schedule.	I think it is an interesting research area. In practice, it is really difficult to fit into a project's schedule. It is worth noticing that this research does not consider all factors but focuses on analysis only; however, analysis is one of the most important areas.

Table 8
Expert opinions regarding RQ3 (impact of documentation technique on testing effectiveness)

Practitioner 1	**Practitioner 2**
Q1.1: Is the purpose and design of the conducted research study comprehensible?	
Yes.	Yes.
Q1.2: Is the presentation of the results obtained in this study clear and unambiguous?	
Yes.	Yes.
Q1.3: Are the results obtained consistent with your professional experience?	
I am not really able to answer that because, in my job history, I have rarely used detailed use case descriptions.	I agree that preparing use case specifications helps testers in their tasks, but at the same time it is time- and effort-consuming.

Table 8 (cont.)

Practitioner 1	**Practitioner 2**
Q1.4: Do you consider these results to be useful for IT projects practitioners?	
I believe these results will be useful to senior analysts who are in charge of analysis activities in a project and have to select the documentation techniques to be used in such a project. It is important to know that the way requirements are documented influences the work of other project participants (in this case, testers).	In the projects for which I have worked, I often encountered the problem of insufficient time to thoroughly test a product, which later resulted in increased numbers of bug reports from customers. Despite the fact that specifying use cases is expensive, after learning about this study's results, I think I will try to introduce it to my current team. I believe further studies on this matter would be beneficial.

This structured part of the interview was followed by a less formal one aimed at discussing the findings and the possibility of generalizing the approach used as well as at sharing ideas on the directions of further research. The results of the informal part of the interviews are incorporated in the discussion of our approach (Section 4) and conclusions (Section 5).

3.6. Threats to validity

We consider the presented study to be an initial attempt, and we are aware of its limitations and threats to validity. Below, we discuss the four aspects of validity following the guidelines for reporting case study research available in [34]:

- **Construct validity** – this is a question to what extent the operational measures that are studied really represent, what the researchers have in mind, and what is investigated according to the research questions. We minimized such threats by applying a systematic GQM approach in order to derive concrete metrics (preferably based on hard numbers) contributing to each research question.
- **Internal validity** – this aspect is important when causal relationships are examined and expresses the risk that another factor not considered by researchers can affect the investigated phenomena. We made an effort to minimize the likelihood of such a situation by selecting projects of similar product sizes, team sizes, and developmental approaches. Despite this effort, we cannot claim that the relationships we identified (e.g., between the requirement elicitation approach and elicited requirement quality) were completely free from other influencing factors. A real-life software project is a very complex enterprise; contrary to controlled experiments, it is simply not possible to organize projects in a manner that ensures that they are shielded from any factors except for a small number of independent variables. This is the threat we accept and acknowledge.

- **External validity** – this aspect questions to what extent it is possible to generalize the findings outside the specific investigated case. Our study was conducted using the data from a single company, which does not allow us to reach strong conclusions and generalize our results. The company and its products are not very unique nor strikingly different from the industrial practice, but they cannot be considered to be representative of the industry in general (e.g., for other types of products, from entertainment software to safety-critical control systems). Even within the context of this single company, our results are not entirely generalizable, as we analyzed a low number of projects similar to each other. This, however, was a trade-off to mitigate the other (internal validity) threat: attempting to establish dependencies and causal relationships in the presence of other significantly influencing factors.
- **Reliability** – is the concern to what extent the data and the analysis are dependent on the specific researchers. All of the steps of the presented study as well as the detailed activities were planned in advance in a consensus-based manner, which minimized the threat of bias being introduced by a single person. As the study was based on data from software supporting tools, most of the data (including all of the quantitative data) was directly extracted from these tools and did not require any additional analysis. An exception is the classification of requirement errors (presented in Section 3.3), as it may sometimes be difficult to pinpoint the exact defect category and decide whether such a requirement is ambiguous or rather incomplete, for example. It was merely an extended analysis of the relationship between requirement elicitation techniques and the resulting requirement quality. Determining the relationship itself did not require any of our decisions – erroneous requirements were pointed out by the project participants, not by us.

4. Discussion of our approach

As presented in the previous section, the data stored in the Jira and Confluence tools as part of the usual development activities (without any additional reporting) could be used to investigate some aspects of the impact of the different RE practices used among the considered projects. We presented a single study that we conducted, but we believe it can be generalized; i.e., similar analyses can be applied to the data from other projects supported by such tools. Jira is one of the tools most widely used in software projects, especially those that apply Agile methods or at least some Agile practices (like the company presented in this paper). The results of the most recent Version One survey [40] announced Jira to be the most popular Agile management tool. Confluence is a team-collaboration tool offered by the same manufacturer and easily integrated with Jira. It is clear that an evaluation approach dedicated to such a toolset is likely to have widespread application.

The case study demonstrated that it is possible to explore various correlations and investigate any possible causal relationships, even despite the fact that we had no

influence on the processes and work procedures of the considered projects – we merely received data from those projects that had already been finished. We could not even plan a priori the exact research questions, as we were dependent on the contents of the received dataset. Such datasets should be considered as a potentially rich source of information. Various kinds of research studies can take advantage of them; this can serve as a post-mortem analysis (using the data from concluded projects, as we did), action research (where the outcomes of decisions made in an ongoing project are investigated), or even controlled experiments (which, however, are unlikely in an industrial setting, as discussed in Section 1). The most promising direction (with respect to the value provided) would be to apply our approach to ongoing and future projects for the purpose of predicting and/or monitoring a project. In the cases of larger samples, the statistical significance can be checked by the appropriate tests (which we omitted due to the small size of our data).

If a research study is planned for an ongoing or future project, some analyses can be easier to conduct if an additional piece of information is registered. One should be careful not to place the burden of additional reporting on project participants, as it contradicts the principles of Agile experimentation. However, the addition of a single data field can sometimes make a huge difference. In our case, the explicit categorization of each discovered requirement defect (completeness, unambiguity, testability, etc.) would significantly reduce the effort necessary to obtain the information presented in Table 2 and Fig. 2. If such a categorization was given when submitting a defect report, the effort would drop from the many hours we spent on classifying the defects to zero. It is also quite likely that such a field would help analysts understand any reported problems and introduce their corrections. This would require a change in reporting practices and in the software tools; however, Jira is known to be a very configurable tool.

Some information is unlikely to be extracted from the datasets exported from tools. In our case, such missing information included requirement elicitation techniques, for example. Each requirement stored in Jira had a source (a stakeholder) assigned, but there was no information about the RE technique used to elicit a given requirement; from our knowledge, this is a rather typical situation. Fortunately, we were able to gather the information we needed (including elicitation techniques) from the people involved in software projects. Access to such people is usually possible when researchers cooperate with a company; this is even more likely if the company is investigating its own data.

To facilitate such analyses, Jira could be extended with a dedicated plug-in. We reviewed the Atlassian Marketplace[3] website, which lists more than 1500 Jira plug-ins; however, we found no such plug-in. This seems to be a niche worth exploring.

[3]https://marketplace.atlassian.com/addons/app/jira

5. Conclusions

In this paper, we discussed the feasibility of a lightweight approach to analyzing the impact of RE practices. From a literature review, we identified that few similar research studies had been published and that the cost of this kind of research (the work effort required) is probably the main obstacle preventing its wider adoption. Inspired by the ideas of Agile experimentation, we proposed an approach based on using popular tools and data registered in them during their standard usage in a software project. We conducted an initial case study by analyzing the data from six similar projects, and we were able to show some potentially interesting dependencies by comparing the measurements of projects that used different RE practices; namely, requirement quality, RE rework effort, and testing efficiency. Despite our study's limitations, we consider it to be a proof of concept that such a lightweight approach to investigating RE-related phenomena is feasible, and we also believe that other project areas (and their related measurements) can be selected; e.g., architectural design or development (coding). We outlined our idea, included the lessons learned from the conducted study, and identified some of the factors facilitating the adoption of our proposal.

Implications for researchers: this paper identified a research gap – the lack of a feasible low-cost approach to investigate and measure the impact of RE on other project areas. We presented a proposal of such an approach and the results from the initial case study, which are, however, not a definite argument due to the study's limitations and the validity threats. Therefore, more research in this area is needed to be able to confirm or contradict our findings and to understand the broader range of the effects that RE practices and techniques can have on the various areas and activities of an IT project.

Implications for practitioners: we were able to demonstrate our findings related to three dependencies between RE practices and some results of their applications. Despite the limitations of these findings, they can be considered by practitioners when making decisions regarding the requirement elicitation and requirement documentation techniques to be used in a given project (or at least expert opinions gathered during validation indicate that it could be worthwhile). Also, these findings can be used to argue the importance of requirement engineering to decision-makers (managers, project sponsors, etc.), as RE activities unfortunately tend to be occasionally neglected in practice.

The most natural direction of any future work is to conduct more case studies and gather a larger and more diversified dataset. We are currently negotiating with two other companies to gain access to the databases of their tools. This would provide us with a significantly larger set of software projects, including more-complex ones that record their activities in the form of several thousands Jira issues, for example. A further step that we consider is the design and implementation of a Jira plug-in to support analyses similar to those described in this paper. However, this would require decisions about particular metrics to be computed, and we believe it should be

preceded by identifying the questions and metrics considered most useful to analysts, project managers, and other industry practitioners.

References

[1] Ambroziewicz A., Śmiałek M.: Applying Use Case Logic Patterns in Practice: Lessons Learnt. In: *Proc. of 20th KKIO Software Engineering Conference, Engineering Software Systems: Research and Praxis, AISC series vol. 830*, pp. 34–49. Springer, 2018.

[2] Bjarnason E., Wnuk K., Regnell B.: Requirements are slipping through the gaps - A case study on causes & effects of communication gaps in large-scale software development. In: *2011 IEEE 19th international requirements engineering conference*, pp. 37–46. IEEE, 2011.

[3] Bormane L., Gržibovska J., Bērziša S., Grabis J.: Impact of requirements elicitation processes on success of information system development projects. In: *Information Technology and Management Science*, vol. 19(1), pp. 57–64, 2016.

[4] Broy M.: Requirements engineering as a key to holistic software quality. In: *International Symposium on Computer and Information Sciences*, pp. 24–34. Springer, 2006.

[5] Carrizo D., Dieste O., Juristo N.: Systematizing requirements elicitation technique selection. In: *Information and Software Technology*, vol. 56(6), pp. 644–669, 2014.

[6] Cheng B.H., Atlee J.M.: Research directions in requirements engineering. In: *2007 Future of Software Engineering*, pp. 285–303. IEEE Computer Society, 2007.

[7] Damian D., Chisan J.: An empirical study of the complex relationships between requirements engineering processes and other processes that lead to payoffs in productivity, quality, and risk management. In: *IEEE Transactions on Software Engineering*, vol. 32(7), pp. 433–453, 2006.

[8] Ellis K., Berry D.M.: Quantifying the impact of requirements definition and management process maturity on project outcome in large business application development. In: *Requirements Engineering*, vol. 18(3), pp. 223–249, 2013.

[9] Gorschek T., Davis A.M.: Requirements engineering: In search of the dependent variables. In: *Information and Software Technology*, vol. 50(1-2), pp. 67–75, 2008.

[10] IEEE: IEEE Standard 830-1998: IEEE Recommended Practice for Software Requirements Specifications, 1998.

[11] IIBA: A Guide to the Business Analysis Body of Knowledge (BABOK Guide) version 3, 2015.

[12] IREB: Handbook of Advanced Level Elicitation according to the IREB Standard, 2019.

[13] ISO/IEC/IEE: ISO/IEC/IEEE Standard 29148-2011. Systems and Software Engineering - Life Cycle Processes - Requirements Engineering, 2011.

[14] Jarzębowicz A., Poniatowska K.: *Towards a Lightweight Approach for the Evaluation of Requirements Engineering Impact on Other IT Project Areas*, pp. 171–186. Springer International Publishing, Cham, 2020. http://dx.doi.org/10.1007/978-3-030-26574-8_13.

[15] Jarzębowicz A., Ślesiński W.: What is Troubling IT Analysts? A Survey Report from Poland on Requirements-related Problems. In: *Proc. of 20th KKIO Software Engineering Conference, Engineering Software Systems: Research and Praxis, AISC series vol. 830*, pp. 3–19. Springer, 2018.

[16] Juristo N., Moreno A.M.: *Basics of software engineering experimentation*. Springer Science & Business Media, 2013.

[17] Kamata M.I., Tamai T.: How does requirements quality relate to project success or failure? In: *15th IEEE International Requirements Engineering Conference (RE 2007)*, pp. 69–78. IEEE, 2007.

[18] Khan H., Asghar I., Ghayyur S., Raza M.: An empirical study of software requirements verification and validation techniques along their mitigation strategies. In: *Asian Journal of Computer and Information Systems*, vol. 3(03), 2015.

[19] Kopczyńska S., Nawrocki J., Ochodek M.: An empirical study on catalog of non-functional requirement templates: Usefulness and maintenance issues. In: *Information and Software Technology*, vol. 103, pp. 75–91, 2018.

[20] Lethbridge T.C., Lyon S., Perry P.: The Management of University–Industry Collaborations Involving Empirical Studies of Software Enginee. In: *Guide to Advanced Empirical Software Engineering*, pp. 257–281. Springer, 2008.

[21] Liechti O., Pasquier J., Reis R.: Beyond dashboards: on the many facets of metrics and feedback in agile organizations. In: *2017 IEEE/ACM 10th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pp. 16–22. IEEE, 2017.

[22] Maalem S., Zarour N.: Challenge of validation in requirements engineering. In: *Journal of Innovation in Digital Ecosystems*, vol. 3(1), pp. 15–21, 2016.

[23] Madeyski L., Kawalerowicz M.: Software engineering needs agile experimentation: a new practice and supporting tool. In: *Proc. of 18th KKIO Software Engineering Conference, Software Engineering: Challenges and Solutions, AISC series vol. 504*, pp. 149–162. Springer, 2017.

[24] McManus J., Wood-Harper T.: Understanding the Sources of Information Systems Project Failure - A study in IS project failure. In: *Management Services Journal*, vol. 51, pp. 38–43, 2007.

[25] Méndez Fernández D., Mund J., Femmer H., Vetrò A.: In quest for requirements engineering oracles: dependent variables and measurements for (good) RE. In: *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, p. 3. ACM, 2014.

[26] Méndez Fernández D., Wagner S., Kalinowski M., Felderer M., Mafra P., Vetrò A., Conte T., Christiansson M.T., Greer D., Lassenius C., et al.: Naming the pain in requirements engineering: Contemporary problems, causes, and effects in practice. In: *Empirical software engineering*, vol. 22(5), pp. 2298–2338, 2017.

[27] Mossakowska K., Jarzębowicz A.: A Survey Investigating the Influence of Business Analysis Techniques on Software Quality Characteristics. In: *Proc. of 19th KKIO Software Engineering Conference, Towards a Synergistic Combination of Research and Practice in Software Engineering, SCI series vol. 733*, pp. 135–148. Springer, 2018.

[28] de Oliveira Neto F.G., Horkoff J., Knauss E., Kasauli R., Liebel G.: Challenges of aligning requirements engineering and system testing in large-scale agile: A multiple case study. In: *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, pp. 315–322. IEEE, 2017.

[29] PMI: Business Analysis for Practitioners A Practice Guide, 2015.

[30] Przybyłek A., Kowalski W.: Utilizing online collaborative games to facilitate agile software development. In: *2018 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 811–815. IEEE, 2018.

[31] Radliński Ł.: Empirical analysis of the impact of requirements engineering on software quality. In: *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pp. 232–238. Springer, 2012.

[32] Rapp D., Hess A., Seyff N., Spörri P., Fuchs E., Glinz M.: Lightweight requirements engineering assessments in software projects. In: *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, pp. 354–363. IEEE, 2014.

[33] REQB: REQB CPRE Foundation Level Syllabus ver. 2.1, 2014.

[34] Runeson P., Höst M.: Guidelines for conducting and reporting case study research in software engineering. In: *Empirical software engineering*, vol. 14(2), p. 131, 2009.

[35] Sethia N.K., Pillai A.S.: The effects of requirements elicitation issues on software project performance: An empirical analysis. In: *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pp. 285–300. Springer, 2014.

[36] van Solingen D.R., Berghout E.W.: *The Goal/Question/Metric Method: a practical guide for quality improvement of software development.* McGraw-Hill, 1999.

[37] Sommerville I., Ransom J.: An empirical study of industrial requirements engineering process assessment and improvement. In: *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 14(1), pp. 85–117, 2005.

[38] The Standish Group: Chaos Report available:, 2014. https://www.projectsmart.co.uk/white-papers/chaos-report.pdf.

[39] Verner J., Cox K., Bleistein S., Cerpa N.: Requirements engineering and software project success: an industrial survey in Australia and the US. In: *Australasian Journal of information systems*, vol. 13(1), 2005.

[40] Version One: 12th Annual State of Agile Report, 2018. https://explore.versionone.com/state-of-agile/versionone-12th-annual-state-of-agile-report.

[41] Wellsandt S., Hribernik K.A., Thoben K.D.: Qualitative comparison of requirements elicitation techniques that are used to collect feedback information about product use. In: *Procedia CIRP*, vol. 21, pp. 212–217, 2014.

[42] Wieringa R., Maiden N., Mead N., Rolland C.: Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. In: *Requirements engineering*, vol. 11(1), pp. 102–107, 2006.

[43] Zhang Z.: Effective requirements development-a comparison of requirements elicitation techniques. In: *Software Quality Management XV: Software Quality in the Knowledge Society, E. Berki, J. Nummenmaa, I. Sunley, M. Ross and G. Staples (Ed.) British Computer Society*, pp. 225–240, 2007.

Affiliations

Aleksander Jarzębowicz
Department of Software Engineering, Faculty of Electronics, Telecommunications, and Informatics, Gdańsk University of Technology, Gdańsk, Poland, email: olek@eti.pg.edu.pl, ORCID ID: https://orcid.org/0000-0003-3181-4210

Katarzyna Poniatowska
Department of Software Engineering, Faculty of Electronics, Telecommunications, and Informatics, Gdańsk University of Technology, Gdańsk, Poland, email: katarzynaponiatowska94@gmail.com

Received: ??.??.20??
Revised: ??.??.20??
Accepted: ??.??.20??