Postprint of: Jarzębowicz A., Poniatowska K. (2020) Towards a Lightweight Approach for the Evaluation of Requirements Engineering Impact on Other IT Project Areas. In: Jarzabek S., Poniszewska-Marańda A., Madeyski L. (eds) Integrating Research and Practice in Software Engineering. Studies in Computational Intelligence, vol 851. Springer, Cham. https://doi.org/10.1007/978-3-030-26574-8_13

Towards a Lightweight Approach for the Evaluation of Requirements Engineering Impact on Other IT Project Areas

Aleksander Jarzębowicz and Katarzyna Poniatowska

Abstract Requirements Engineering (RE) is recognized as one of the most important, but difficult areas of software engineering, with a significant impact on other areas of the IT project and its final outcome. The empirical studies investigating this impact are hard to conduct, mainly due to large effort required. It is thus difficult for researchers and even more for industry practitioners to make evidence-based evaluations, how decisions about RE (e.g. RE process improvements, RE techniques selection) translate into requirements quality and influence other project areas. We propose an idea of a lightweight approach, utilizing the popular tools adopted by numerous software companies, to enable such evaluation without an excessive effort. The proposal is illustrated with a pilot study, where the data from 6 industrial projects from a single organization was analyzed and 3 metrics regarding requirements quality, rework effort and testing were used to demonstrate the impact of different RE techniques applied among considered projects. We also discuss the factors important to enabling adoption of the proposed approach.

Key words: Requirements Engineering; Impact; Evaluation; Case study

1 Introduction

Requirements Engineering (RE) is one of the most important, but also difficult areas of software engineering, as many RE-related problems and challenges are still reported by practitioners [1]. It is also known to have a significant impact on other

Aleksander Jarzębowicz (corresponding author, ORCID: 0000-0003-3181-4210)
Department of Software Engineering, Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Gdańsk, Poland, e-mail: olek@eti.pg.edu.pl

Katarzyna Poniatowska
Department of Software Engineering, Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Gdańsk, Poland

areas of the IT project and its final outcome - numerous sources e.g. [2][3][4] describe the relationships between RE-related problems and project failures or challenges.

However, when it comes to empirical studies investigating RE impact i.e. the consequences of decisions about RE process, applied techniques, practices followed etc. there are not so 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. [5][6][7]. The scope of such research studies is usually limited though, as their purpose is to apply and evaluate the single proposal under consideration. Only a few sources that attempt to analyze a wider set of RE process variables and their outcome in an industrial setting are available.

The possible reasons for such lack of empirical studies and quantitative data include the difficulty to measure or manipulate essential variables and a very high effort required [8]. Some examples that illustrate the scale of such effort include: 18-month project involving 9 companies [9]; 30-month project in a single company requiring a serious commitment of researchers [10]; or two large survey studies gathering data from over 400 organizations [11].

It is thus difficult for researchers and even more for industry practitioners to make evidence-based evaluations that investigate how decisions about RE translate into requirements quality and influence other project areas. On the other hand, such evaluations can provide valuable information, both about the more generic picture of RE practices and their effects among IT industry and about the opportunities of improvement for a particular company. By the latter we mean focusing on the specific context and processes of the company and using results of 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), which runs commercial projects, would dedicate significant resources to such investigations, much less to conduct experiments related to their development processes [12][13]. This situation could change though, if the means to conduct such evaluations in a way not interfering with project tasks and not requiring much effort were provided.

We tried to determine, if such means can be provided by utilizing the existing (and popular) approaches and tools. For that reason we conducted a case study and analyzed data from 6 concluded projects from a single software company. From the data and additional background knowledge provided by people involved in those project, we were able to determine that the application of different RE practices translated into differences in metrics reflecting requirements' quality, rework effort and testing effectiveness.

Our work draws upon the ideas of agile experimentation [14], which proposes small-scale experimenting in real-life industrial projects. Such approach is expected to provide more meaningful results than e.g. classroom experiments and at the same time to minimize the risk that experimenting may impede the main goal (delivering the software to the customer) or significantly increase the workload of project participants. The research described in this paper can hardly be called experimentation – the pilot study we conducted was an ex-post analysis of the data

from past projects, however we can see the opportunity to use the proposed approach in the ongoing or planned projects, for ex-ante predictions.

The main contributions of this paper are: (1) the industrial case study investigating the impact of RE practices on other project areas and (2) the proposal of lightweight approach to evaluate RE impact in industrial projects, including experimentation in ongoing projects.

The paper is organized as follows. Section 2 provides more background by discussing the related work. In Section 3 the industrial case study based on 6 projects and its findings is described. In Section 4, we attempt to generalize the described case study into universal approach enabling the evaluation of RE impact and experimentation in an industrial setting. The paper is concluded in Section 5.

2 Related Work

The work related to our research encompasses the broader empirical studies focusing on RE and its impact. By broader we mean those that take into consideration several variables, as opposed to the 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) ones more similar to our work e.g. case studies, action research or controlled experiments, but also (2) questionnaire surveys or interviews. Below we summarize the research belonging to each of these two groups.

Sommerville and Ransom [9] conducted a study during a project from EU framework program. Its main purpose was to improve RE processes of 9 software companies belonging to project consortium. The metrics collected by them shown the improvement (of varying degree) in all companies as well as better business performance. Damian and Chisan [10] focused on RE processes of a single organization during a 30-month study. They analyzed the processes, suggested improving practices and evaluated the outcome, noticing a positive impact on other project areas as well as on general productivity, quality and risk management. Kamata and Tamai [15] investigated the correlation between the completeness and quality of software requirements specifications (SRS) and project success (in terms of schedule and budget). For that reason they reviewed 32 SRS documents using IEEE 830:1998 [16] standard as a reference point. Results showed that successful projects had more balanced coverage of various requirements categories and that “purpose”, “product perspective” and “functions” sections of SRS were crucial. Rapp et al. [17] developed a questionnaire-based method of RE processes assessment and validated it by assessing several industrial and semi-industrial projects. They also stressed that such method must minimize assessment effort and provide a real business value to assessors and organizations undergoing assessments. Bormane et al. [18] analyzed 12 software projects from a major Latvian company, comparing waterfall/agile and small/large projects. They found agile approach to require much lower cost of changes and to provide more accurate effort and budget estimations in case of large projects.

They also analyzed elicitation techniques used in waterfall and agile projects but were not able to draw conclusions from the dataset used.

Verner et al. [19] surveyed practitioners from Australia and U.S., gathering data from 164 projects. Their findings indicate that effective requirements elicitation and requirements management are the most important contributing factors to project success. Bjarnason et al. [20] performed an interview-based exploratory case study investigating gaps in communication of requirements. Their study revealed (among other things) the consequences of such gaps that affect e.g. requirements quality, failure to meet stakeholders expectations, testing efficiency. Ellis and Berry [11] 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 the RE maturity model, while the second one demonstrated a positive correlation between RE maturity and the ratio of successfully completed strategic projects. Mossakowska and Jarzębowicz [21] conducted a survey aimed at identifying which RE techniques are perceived by practitioners as contributing to particular quality characteristics of the final software product.

In general, there are not many sources that could be considered a directly related work. Some of the studies on RE impact are rather based on surveys and interviews than on quantitative analysis of ongoing or concluded software projects. Among the remaining ones, most studies required large workload to conduct, which contradicts our postulate to minimize the effort. The studies most similar to ours are [15] and [18] , which analyze the data from concluded projects and seek correlations between RE and outcomes measured at the level of the whole project. We however 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 software company we approached. This section provides the background information about the company and its projects, describes the design of our study and reports the results obtained.

3.1 Background

The research was conducted in a medium Polish company employing about 200 people. It specializes in development of electronic measuring devices (dedicated to various utilities like water, heat, gas etc.) and related software systems (web and mobile applications for configuration and administration of devices, reading the

measurements through e.g. radio channel and processing measurement data). The customers of such software are both internal and external, the external ones include domestic and foreign organizations.

Software development projects run by the company differ with respect to their size and methodology used, which led us to narrowing our focus to more specific type of projects we included in our study. The following description of development processes is also narrowed down to this type of projects. Such projects are developed in small teams (6-9 people), including 1 or 2 analysts responsible for requirements and used a hybrid development approach, based on incremental software lifecycle model and integrating mostly plan-driven and some agile practices. This approach was most common for projects in recent years as it was introduced by the company which intended to introduce agile practices to some extent, but not attempted to make a full transition to agile development methods.

This hybrid development process can be summarized as follows. The analysis phase is to large extent completed at the beginning of the project, and results in capturing its 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 about the component under consideration. Such document is called a detailed software requirements specification. It uses a standardized template, based on IEEE 830:1998 [16] (this standard was superseded by ISO/IEC/IEEE 29148 [22] a few years ago, but is still used as a point of reference by the company). The template was developed by the company for the internal use in all 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 requirements i.e. whether they are unambiguous, testable etc. After reviews and possible corrections, the requirements are passed to the developers, who work on the code, when necessary contacting the analysts to clarify doubts. When the component is ready, it is passed to testers who verify its quality. It is followed by a phase of fixing defects and stabilizing, resulting in concluding the work on the component. Another increment dedicated to another component and following the same workflow starts subsequently. Such process continues until the whole product is ready, then a delivery to the customer takes place. The agile practices are mostly related to the work of the development team and include self-organizing teams, prioritized work list, daily stand-up meetings and retrospectives.

Software projects run by the company are supported by integrated Confluence[1] and Jira[2] tools (both provided by Atlassian) as well as associated time-tracking plugin. Confluence is a collaboration software, based on the idea of wiki. 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

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

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

fragments of the specification. The company's policy is to register every such query related to requirements. This, in addition to the versioning mechanism provided by Confluence, allows to track the evolution of requirements specification including fixing 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 software projects of the discussed company, Jira is applied for both purposes. Apart from the typical usage of defining issues representing product features and development tasks (plus updating their statuses and reporting work effort), also requirements engineering activities are strongly supported by this tool. The project manager defines an issue corresponding to task of software requirements specification development and analysts report the work effort dedicated to this task by registering the particular activities they conducted together with time spent on each of them. 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 obliged to report the time spent on it. Such information is registered and associated with this issue. When requirements specification is ready to be verified, the analyst creates new issues dedicated to reviews and assigns them to 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 already 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 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 e.g. size or development methodology) as such differences could possibly obscure the phenomenon we investigated.

We selected 6 software projects conducted within 2014-2017 period. 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, clearly some differences can be noticed, but they do not exceed 25% (and it is hard to expect equal values from real life examples).

More significant differences concern requirements, in particular the techniques of requirements elicitation and requirements documentation. "Req. elicitation" row of Table 1 enumerates elicitation techniques used by analysts in a given project. Most of them are well-known techniques, one explanation is however necessary: SIWZ is the Polish abbreviation of a document type used in public projects. The

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
Elicitation	interviews, prototyping	interviews, observation, prototyping	document analysis (SIWZ), interviews	workshop, observation	document analysis (SIWZ), interface analysis	document analysis (SIWZ), system archaeology
Documentation	user stories, tables	tables, UC	tables	tables, UC	tables	tables
No. of req. defects reported	11	12	16	13	14	21

public organization acting as project customer is obliged by Polish law to publish the request for proposals and the terms of reference. The latter are described in SIWZ document (and its attachments) and can include many details on requirements. 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, description. The description was expressed in natural language, without any internal structure.
- UC – use cases providing a structured description of functional requirements, with detailed interaction scenarios (basic and alternative).
- User stories – a simplified form of expressing requirements: As <role> I want <capability> so that <goal>.

All analyzed projects (P1-P6) used tables to represent requirements, however some of them refined functional requirements to 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 of documenting requirements.

3.3 Study Design

The data from Jira and Confluence can be exported to a spreadsheet format. We received the data from the abovementioned projects in such form and used it to conduct our study. Additionally, we could contact some people involved in those projects, who were able to provide us with background information, not necessarily registered in supporting tools e.g. requirements elicitation techniques applied or stakeholders' attitude.

We started this research with the intent of exploring the quantitative impact of RE on requirements quality and on the work done in other areas of IT project. We had not formulated more specific research questions a priori, because, unlike a controlled experiment, such study was dependent on the data we could obtain. For instance, if the data about requirement-related defects was not available (because company's procedures had not required reporting them or the supporting tool does not allow to distinguish them from other issues), it would be impossible to investigate any consequences of such defects. The scope of reliable information we could gather from people involved in those 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 in case time was not registered by tools, we would not be able to use this metric.

After the initial examination of the data and gathering background information related to investigated projects, we were able to define 3 research questions:

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

Each research question was refined for the purpose of operationalization. We used GQM (Goal-Question-Metric) [23] method to identify 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 particular elicitation techniques used in each project. However, a more fundamental related issue was communicated by project participants. In some of the investigated projects (P3, P5, P6) the main source of requirements was the document prepared by the customer (SIWZ) and the communication with customer representatives was very limited (by phone only, moreover stakeholders were often reluctant to cooperate and required significant communication effort from the analyst). In those project the most common technique applied was document analysis. Additionally analysts used other techniques like interface analysis or system archaeology (on competitors' systems), that could provide them the information that was hard to get directly from customer representatives. In the remaining projects (P1, P2, P4) the analysts were able to act more actively and to meet customer representatives, organize requirements elicitation sessions and get to know the environment the prospective users work in. The stakeholders were rather cooperative and techniques like interviews, workshops, prototyping and observation were used. We decided to distinguish such two general approaches to requirements elicitation, named "active" (P1, P2, P4) and "passive" (P3, P5, P6) and to investigate

whether any difference between the quality of requirements elicited by them could be noticed.

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

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

We verified that the data exported from the tools allowed to trace the comments containing defect suggestions, doubts, questions etc. associated with particular requirements and the history of changes made to requirements. Additionally, we decided to categorize the defects related to requirements, adopting the classification from [22] (section 5.2.5) and [24] (section 7.2.4). This task had to be done manually though, because no such categorization was used when reporting defects in the considered projects.

3.3.2 RQ2

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

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

3.3.3 RQ3

In this research question we intend to find out whether different requirements documentation techniques affect the work of testers. As shown in Table 1, 3 techniques were used in P1-P6: tables, use cases and user stories. Among these 3 techniques,

use cases were much more detailed representation than the remaining ones. We decided to compare projects where use cases were applied and the remaining ones. The rationale was that more detailed requirements specifications can help testers to design more thorough test cases and test scenarios, both positive and negative ones. To quantify the impact on the work of 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 and all software bugs uncovered in the product (those found during internal tests and those reported by customers or other external parties). As products of all considered projects were delivered to customers, stabilized and used for some period of time, such external bug reports were available. We counted all issues assigned to bug reports category and distinguished them according to source (internal/external). The GQM structure for this part of our research was as follows:

- **Goal**: Assess the impact of requirements documentation technique on the effectiveness of testing.
- **Question**: What is the difference in the effectiveness of testing between the projects which applied use cases and other projects?
- **Metric**: The proportion of bugs found during the internal testing to and 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 to categorize them. Most of the defects were reported before implementation of requirements (e.g. developers asking for explanations of ambiguous requirements, testers questioning requirements testability), but there was a minority of defects registered after implementation (e.g. a feedback from a business stakeholder about wrong software feature, which was traced back to incorrect requirement). The summary of our findings is shown in Table 2. Additional visualization is provided in Fig. 1, which depicts the percentage of erroneous requirements and the distribution of particular kinds of defects. An observation can be made, that projects where active approach to requirements elicitation was used, have relatively fewer defects.

The most frequent kind of defects concerns unambiguity, in both “active” and “passive” projects, but the former are not as affected as the latter. The possible explanation is that active approach forces the analyst to ensure that the requirements he/she elicits from a stakeholder are mutually understood in the same way. On the other hand, 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 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 active approach resulted in fewer defects concerning feasibility. This issue was discussed by us with a group of analysts (of

Table 2 Requirement defects in analyzed projects.

Project	P1	P2	P3	P4	P5	P6	Total
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	
Defect categorization:							
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

11, 7 and 5 years of experience respectively). The discussion led to the conclusion that document analysis elicitation technique can possibly decrease the awareness of the analyst – he/she can make an unfounded assumption regarding feasibility (that if a requirement was included in an official document then it can be implemented) and refrain from consulting it with developers.

Exploration of RQ2 led to juxtaposition of requirements quality and RE rework effort, presented in Table 3. To represent quality, the percentage of erroneous requirements was used, computed on the basis of values from Table 2. The rework is represented by the amount 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 3 we also show the relative value of RE rework com-

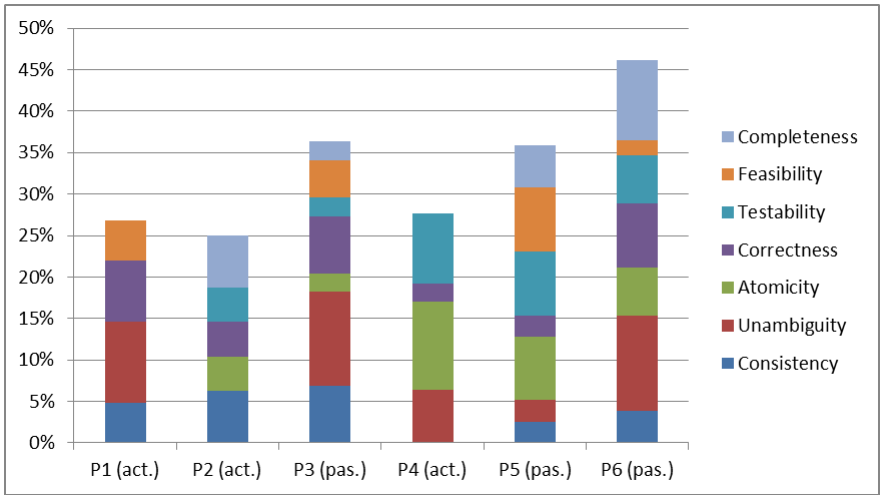

Fig. 1 The percentage of requirements containing defects with defect classification.

Table 3 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%	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%

paring the effort reported by additional personnel to the total effort for RE activities included in project plan.

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

As indicated in the description of RQ3 in Section 3.3, we compared the effectiveness of testing between projects that had functional requirements represented as use cases and the remaining projects. The results, summarized in Table 4, show the difference – projects with use case specifications had over 80% of bugs found by company's testers, while for other projects the effectiveness of testing was about 70%.

Table 4 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 by conducting interviews with 3 testers (who participated in some of the analyzed projects) revealed that they considered use cases as the most helpful requirements documentation technique and criticized the 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 a clarification) and by the fact that use cases

can relatively easy be transformed into test cases. This finding is however based on interviews only, not the quantitative data derived from tools.

3.5 Threats to Validity

We consider the presented study as an initial attempt and we are aware of its limitations and threats to validity.

The study was conducted using the data from a single company, which does not allow to reach strong conclusions and to generalize our results. The company and its products are not very peculiar and strikingly different from the industrial practice, but they cannot be considered representative for 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. It however was a trade-off to mitigate the other threat: attempting to establish dependencies and causal relationships in the presence of other significantly influencing factors. Despite we cannot claim that the relationships we identified (e.g. between requirements elicitation approach and elicited requirements quality) were completely free from other influencing factors, we made an effort to minimize the likelihood of such situation by selecting projects of similar product size, team size and development approach.

4 Discussion of Our Approach

As presented in the previous section, the data stored in 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 different RE practices used among the considered projects. We presented a single study we conducted, but we believe it can be generalized i.e. similar analyses can be applied to data from other projects supported by such tools. Jira is one of the tools most widely used in software project, 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 [25] announce Jira to be the most popular agile management tool. Confluence is a team collaboration tool, offered by the same manufacturer, easily integrated with Jira. It is clear that evaluation approach dedicated for such toolset is likely to have a widespread application.

The case study demonstrated that it is possible to explore various correlations and investigate possible causal relationships, even despite the fact that we had no influence on the processes and work procedures of the considered projects – we just received the data from the projects that had already been finished. We haven't even planned 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, it can be 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 prediction and/or monitoring the project. In cases of larger samples, statistical significance can be checked by appropriate tests, which we omitted because of 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 information is registered. One should be careful not to place the burden of additional reporting on project participants because it contradicts the principles of agile experimentation. However, sometimes addition of a single data field can make a huge difference. In our case, explicit categorization of each requirement defect found (related to completeness, unambiguity, testability etc.) when submitting such defect report would significantly reduce the effort necessary to obtain information presented in Table 2 and Fig. 1 (from many hours spent on classifying defects to none). It is also quite likely that such field would also help analysts to understand reported problems and introduce corrections. This would require change in reporting practices and in 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 it was e.g. requirements elicitation techniques – requirements had a source (a stakeholder) assigned, but it no information about the technique used to elicit a given requirement (nor even about techniques applied in a given project) was available and it is a rather typical situation. We were however able to gather the information we needed from people involved in software projects. Access to such people is usually possible when researchers cooperate with the company, even more so if the company is investigating its own data.

To facilitate such analyses, Jira could be extended with a dedicated plug-in. We reviewed Atlassian Marketplace[3] website which lists over 1500 Jira plug-ins, but we found none of such kind. It seems to be a niche worth exploring.

5 Conclusions

In this paper we discussed the feasibility of lightweight approach to analyzing the impact of RE practices. From the literature review we identified that few similar research studies were published and that the cost of this kind of research (work effort required) is the main obstacle for its wider adoption. Inspired by the ideas of agile experimentation, we proposed an approach based on using the popular tools and data registered in them as result of their "standard" usage in a software project. We con-

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

ducted an initial case study, by analyzing the data from 6 similar projects and we were able to show some potentially interesting dependencies by comparing measurements of projects that used different RE practices, namely: requirements quality, RE rework effort and testing efficiency. Despite study's limitations we consider it a "proof of concept" that such lightweight approach to investigating RE-related phenomena is feasible and we believe that also other project areas (and related measurements) can be selected e.g. architectural design or development (coding). We outlined our idea, also including lessons learned from the conducted study and identifying some of the factors facilitating adoption of our proposal.

The most natural direction of future work is to conduct more case studies and gather a larger and more diversified data. We are currently negotiating with two other companies to get access to the databases of their tools. It would provide us a significantly larger set of software projects, including more complex ones that e.g. record their activities in the form of several thousands of Jira issues. The further step we consider is the design and implementation of Jira plug-in to support the analyses similar to those described in this paper. It would however require decisions about particular metrics to be computed and we believe it should be preceded by gathering more information about the questions and metrics considered most useful to analysts, project managers and other industry practitioners.

References

1. Jarzębowicz, A., Ślesiński, W.: What Is Troubling IT Analysts? A Survey Report from Po-land on Requirements-Related Problems. In Proc. of 20th KKIO Software Engineering Conference, Engineering Software Systems: Research and Praxis, AISC series vol. 830, 3-19, Springer, Cham (2019).
2. McManus, J., Wood-Harper, T.: Understanding the Sources of Information Systems Project Failure - A study in IS project failure. Manag. Serv. 51, 38–43 (2007).
3. The Standish Group: Chaos Report, available: https://www.projectsmart.co.uk/white-papers/chaos-report.pdf (2014).
4. Méndez Fernández, D. et al.: Naming the pain in requirements engineering: Contemporary problems, causes, and effects in practice. Empir. Softw. Eng. 22, 2298–2338, DOI 10.1007/s10664-016-9451-7 (2017).
5. Kopczyńska, S., Nawrocki, J., Ochodek, M.: An empirical study on catalog of non-functional requirement templates: Usefulness and maintenance issues. Information and Software Technology, 103, 75-91 (2018).
6. Przybyłek, A., Kowalski, W.: Utilizing online collaborative games to facilitate agile soft-ware development. In 2018 Federated Conference on Computer Science and Information Systems (FedCSIS), 811-815, IEEE (2018).
7. 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, 34-49, Springer, Cham (2019).
8. Méndez Fernández, D., Mund, J., Femmer, H., Vetrò, A.: In quest for requirements engineering oracles: dependent variables and measurements for (good) RE. In 18th International Conf. on Evaluation and Assessment in Software Engineering (EASE), p. 3, ACM (2014).
9. Sommerville, I., Ransom, J.: An empirical study of industrial requirements engineering process assessment and improvement. ACM Transactions on Software Engineering and Methodology (TOSEM), 14(1), 85-117 (2005).

10. 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. IEEE Trans. Softw. Eng. 32, 433–453 (2006).
11. Ellis, K., Berry, D.M.: Quantifying the impact of requirements definition and management process maturity on project outcome in large business application development. Requir. Eng. 18, 223–249 (2013).
12. Juristo, N., Moreno, A.: Basics of Software Engineering Experimentation, Springer (2010).
13. Lethbridge, T. C., Lyon, S., Perry, P.: The Management of University–Industry Collaborations Involving Empirical Studies of Software Engineering. In: Guide to Advanced Empirical Software Engineering, 257-281, Springer, London (2008).
14. Madeyski, L., Kawalerowicz, M.: Software engineering needs agile experimentation: a new practice and supporting tool. In: 18th KKIO Software Engineering Conference, Software Engineering: Challenges and Solutions, AISC series vol. 504, 149-162, Springer (2017).
15. Kamata, M., Tamai, T.: How does requirements quality relate to project success or failure?. In 15th IEEE International Requirements Engineering Conference, 69-78, IEEE (2007).
16. IEEE: IEEE Standard 830-1998. IEEE Recommended Practice for Software Requirements Specifications (1998).
17. Rapp, D., Hess, A., Seyff, N., Spörri, P., Fuchs, E., Glinz, M.: Lightweight requirements engineering assessments in software projects. In IEEE 22nd International Requirements Engineering Conference, 354-363, IEEE (2014).
18. Bormane, L., Gržibovska, J., Bērziša, S., Grabis, J.: Impact of requirements elicitation processes on success of information system development projects. Information Technology and Management Science, 19(1), 57-64 (2016).
19. Verner, J., Cox, K., Bleistein, S., Cerpa, N.: Requirements engineering and software project success: an industrial survey in Australia and the US. Australasian Journal of information systems, 13(1), 225–238 (2005).
20. 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 19th International Requirements Engineering Conference, 37-46, IEEE (2011).
21. Mossakowska, K., Jarzębowicz, A.: A Survey Investigating the Influence of Business Analysis Techniques on Software Quality Characteristics. In: 19th KKIO Software Engi-neering Conference, Towards a Synergistic Combination of Research and Practice in Software Engineering, SCI series vol. 733, 135-148, Springer, Cham (2018).
22. ISO: ISO/IEC/IEEE Standard 29148-2011. Systems and Software Engineering - Life Cycle Processes - Requirements Engineering (2011).
23. van Solingen, R., Berghout, E.: The Goal/Question/Metric Method: a Practical Guide for Quality Improvement of Software Development, Cambridge University Press (1999).
24. International Institute of Business Analysis: A Guide to the Business Analysis Body of Knowledge (BABOK Guide) version 3 (2015).
25. VersionOne: 12th Annual State of Agile Report, https://explore.versionone.com/state-of-agile/versionone-12th-annual-state-of-agile-report (2018).