Submitted to REFSQ'03 - please do not cite.

Post-Release Analysis of Requirements Selection Quality - An Industrial Case Study

Lena Karlsson[1], Björn Regnell[1], Joachim Karlsson[2], Stefan Olsson[2]

[1]*Department of Communication Systems, Lund University, Sweden {lena.karlsson, bjorn.regnell}@telecom.lth.se*

[2]*Focal Point AB, Linköping, Sweden {joachim.karlsson, stefan.olsson}@focalpoint.se*

Abstract

The process of selecting requirements for a release of a software product is challenging as the decision-making is based on uncertain predictions of issues such as market value and development cost. This paper presents a method aimed at supporting software product development organisations in the identification of process improvement proposals to increase requirements selection quality. The method is based on an in-depth analysis of requirements selection decision outcomes after the release has been launched to the market and is in use by customers. The method is validated in a case study involving real requirements and industrial requirements engineering experts. The case study resulted in a number of process improvement areas relevant to the specific organisation and the method was considered promising by the participating experts.

1 Introduction

This paper presents a method for identifying improvement areas of the requirements selection process in a market-driven software product development context. The method is called PARSEQ (Post-release Analysis of Requirements SElection Quality) and is based on retrospective examination of decision-making in release planning, at a time when the consequences of requirements selection decisions are visible. PARSEQ is applied in a case study where the requirements selection for a particular release of a specific software product is analysed and improvement areas that are relevant to the studied software organisation are identified.

PARSEQ is intended to be used by software organisations that operate in a market-driven context, offering software products to many customers on an open market. Market-driven requirements engineering (RE) differs from customer-specific RE in several ways, for example in the characteristics of stakeholders and schedule constraints [14, 16]. Requirements are often invented by the developers as well as elicited from potential customers with different needs [11], and it is common to use a requirements database that is continuously enlarged with new candidate requirements [5, 12]. Commonly, market-driven software developing organisations provide successive releases of the software product and release planning is an essential activity [1, 2]. A major challenge in market-driven RE is to prioritise and select the right set of requirements to be implemented in the next release [11], while avoiding congestion in the selection process [12]. This decision-making is very challenging as it is based on uncertain predictions of the future, while crucial for the product's success on the market [1, 8].

Given issues such as uncertain estimations of requirements market value and cost of development, it can be assumed that some requirements selection decisions are non-optimal, which in turn may lead to software releases with a set of features that are not competitative or satisfy market expectations. It is only afterwards, when the outcome of the development effort and market value is apparent, it is possible to with more certainty tell which decisions were correct and which decisions were less accurate. But by looking at the decision outcome in retrospect, organisations can gain valuable knowledge of how to improve the requirements selection process and increase the chance of market success.

In [15, 3], post-mortem evaluations are discussed in a project management context. An evaluation of the project's performance after it has been completed is useful both for personal and organisational improvement and can be conducted as an open discussion of the strengths and weaknesses of the project plan and execution. Furthermore, much can be learned about organisational efficiency and effectiveness by this kind of evaluation, which offers an insight into the success or failure of the project. The lessons learned can be used when planning forthcoming projects to improve project performance and prevent mistakes. Continuous process improvement is important in the maturity of software development and, in particular, requirements engineering is pointed out as a critical improvement area in a maturing organisation [10]. A recent process improvement study based on analysis of defects in present products is reported in [9].

The PARSEQ method is evaluated in a case study, where requirements selection decisions for an already released software product were revisited by the decision-makers of the specific organisation [13]. The market value and development cost of the requirements that were candidates for a previous release that was launched 18 months earlier, were re-estimated based on the knowledge gained during the two following releases. The re-estimation resulted in a new priority order, which in turn suggested that some selected requirements should have been postponed and some deferred requirements should have been selected for that release. Each such suspected inappropriate selection was analysed in order to understand the grounds for each decision, which in turn lead to the identification of several areas of process improvements.

The paper is structured as follows. Section 2 presents the PARSEQ methods and its main steps. In Section 3, the case study operation is described and the main results are reported. Section 4 discusses the validity of the findings and the generality of the approach outside the specific case study context. Conclusions and directions of further research are given in Section 5.

2 The PARSEQ Method

Retrospective evaluation of software release planning may give a valuable input to the identification of process improvement proposals. In particular, post-release analysis of the consequences of previous decision-making may be a valuable source of information when finding ways to improve the requirements selection process.

The PARSEQ method is based on a systematic analysis of candidate requirements from previous releases. By identifying and analysing a set of root causes to suspected incorrect requirements selection decisions, it is hopefully possible to find relevant improvements that are important when trying to increase the specific organisation's ability to plan successful software releases.

PARSEQ is divided into 4 steps: requirements sampling, re-estimation of cost and value, root cause analysis, and elicitation of improvements, as shown in Fig 1. The method may use a requirements database as input and assumes that information is available in the database regarding when a requirement is issued and in which release a requirement is implemented. The output of the method is a list of process improvement proposals. Each step in PARSEQ is subsequently described in more detail.

Requirements sampling. The main input to the post-release analysis is a list of requirements that were candidates for a previous product release that now has been out on the market for a time period long enough to allow for an assessment of the current market value of its implemented requirements. First, such a relevant previous release is selected (subsequently called *reference release*). Secondly, the requirements database is examined and those requirements that were candidates for the reference release is retrieved. The previous candidates are requirements that were issued and dated prior to the reference release date, but were not implemented before the reference release, i.e. they were either implemented in the reference release or in a subsequent release, or rejected.

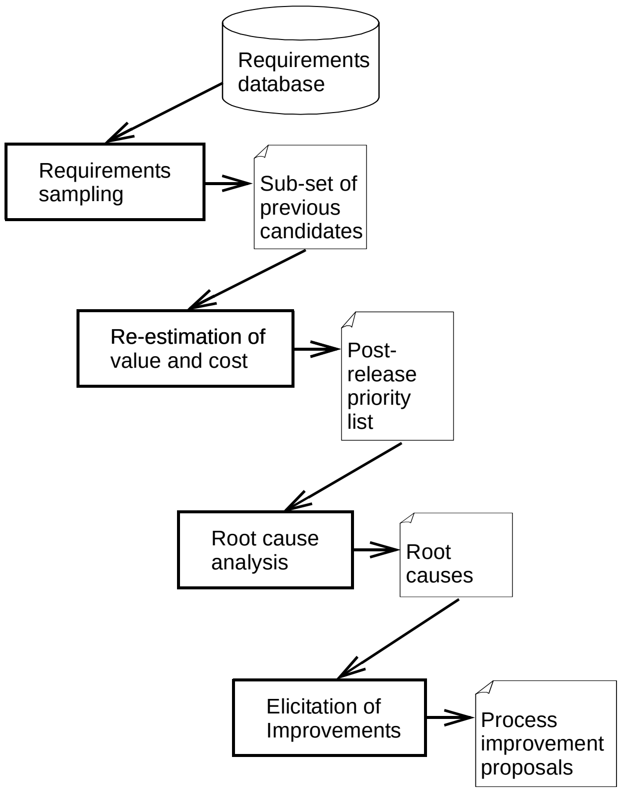
Requirements database
Requirements sampling
Sub-set of previous candidates
Re-estimation of value and cost
Post-release priority list
Root cause analysis
Root causes
Elicitation of Improvements
Process improvement proposals

Fig. 1. An outline of the activities and products of the PARSEQ method.

The purpose of the sampling is to compose a reasonably small but representative sub-set of requirements, since the complete database may be too large to investigate in the post-release analysis. The sample should include requirements that were selected for implementation in the reference release as well as postponed or rejected requirements. The requirement set is thereby useful for the analysis as it consists of typical examples of release planning decisions.

The output from the requirements sampling is a reasonable amount of requirements, high enough to be representative, yet low enough to allow the following steps of PARSEQ to be completed within reasonable time.

Re-estimation of value and cost. The requirement sample is input to the next step of PARSEQ, where a re-estimation of current market value and actual development cost is made in order to find suspected inappropriate decisions that can be further analysed. As the reference release has been out on the market for a while, a new assessment can be made, which applies the knowledge gained after the reference release was launched, which presumably should result in more accurate priorities. The re-estimation is made to find out how the organisation had decided for the reference release, i.e. which requirements that would have been selected, if they knew then what they know now. With todays knowledge, about market expectations and development costs, a different set of requirements may have been selected for implementation in the reference release. If this is not the case, either the organisation has not learned anything since the planning of the reference release, or the market has not changed at all.

The implemented requirements have a known development cost (assuming that outcome of the actual implementation effort is measured for each requirement), but the postponed or rejected requirements need to be re-estimated based on the eventual architectural decisions and the knowledge gained from the actual design of the subsequent releases.

By using, for example, a cost-value prioritisation approach with pairwise comparisons [6, 7], an ordered priority list can be obtained where the requirements with a higher market value and a lower cost of development are sorted in the priority order list before the requirements with a lower market value combined with a higher development cost.

The purpose of the re-estimation is to apply the knowledge that has been gained since the product was released, to discover decisions that would have been made differently today. The discrepancies between the decisions made in the planning of the reference release and the post-release prioritisation is noted and used in the root cause analysis. The output of this step is thus a list of requirements that was given a high post-release priority but were not implemented in the reference release, as well as requirements with a low post-release priority but still implemented in the reference release.

Root cause analysis. The purpose of the root cause analysis is to understand on what grounds release-planning decisions are made. By discussing the decisions made in prior releases, it may be possible to create a basis for the elicitation of process improvement proposals.

The output of the re-estimation, i.e. the discrepancies between the post-release prioritisation and what was actually selected for implementation in the reference release, is analysed in order to find root causes to the suspected inappropriate decisions. This analysis is based on a discussion with persons involved in the requirements selection process. The following questions can be used to stimulate the discussion and provoke insights into the reasons behind the decisions:

- Why was the decision made?
- Based on what facts was the decision made?
- What has changed since the decision was made?
- When was the decision made?
- Was it a correct or incorrect decision?

Guided by these questions, categories of decision root causes are developed. Each requirement is mapped to one or several of these categories to illustrate the decision disposition. This mapping of requirements to root cause categories is the main output of this step together with the insights gained from the retrospective reflection.

Elicitation of improvements. The outcome of the root cause analysis is used to facilitate the elicitation of improvement proposals. The objective of this last step of PARSEQ is to arrive at a relevant list of high-priority areas of improvement. The intention is to base the discussion on strengths and weaknesses of the requirements selection process and to identify changes to current practice that can be realised. The following questions can assist to keep focus on improvement possibilities:

- How could we have improved the decision-making?
- What would have been needed to make a better decision?
- Which changes to the current practices can be made to improve requirements selection in the future?

The results of PARSEQ can then be used in a situated process improvement programme where process changes are designed, introduced and evaluated. These activities are, however, out of the scope of the presented method.

3 Case Study

PARSEQ was tried out in a case study to investigate its feasibility and gain more knowledge for future research on post-release analysis of requirements selection as a vehicle for process improvement. In the first section of this chapter, the case study site and context is described as well as the tool used in the study. Next, the realisation of the PARSEQ method is described, i.e. how each step of the method was carried out in the case study. Finally, the results from the case study are reported, including a number of improvement proposals.

3.1 Background

The case study site is a small-sized organisation developing stand alone software packages. The organisation stores the requirements on the software package in a database that contains already implemented requirements as well as suggestions for new requirements. Each requirement is tagged with a certain state to describe its level of refinement. Examples of states include New, Accepted for prioritisation, Accepted for implementation and Done, see Fig. 2. When a requirement for some reason is not appropriate for the package, its state is set to Rejected. Other states include Clarification needed, Insignificant improvement, Badly documented, Duplicate and Draft.

To analyse the requirements in the database a commercial tool for product management and requirements management, Focal Point[1] was applied. Focal Point has capabilities for eliciting, reviewing, structuring, and prior-

1. For more information see www.focalpoint.se.

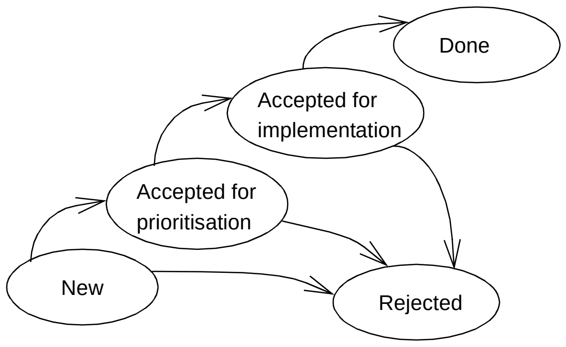

Fig. 2. A simplified version of the requirement state model in the database.

itising requirements as well as for planning optimal releases that maximise the value for the most important customers in relation to development time and available resources. One prioritisation method in Focal Point is pair-wise comparisons [6]. It is helpful for keeping up concentration and objectivity and Focal Point also provides solutions for reducing the number of comparisons and motivating the priorities. This tool also aids in visualising the decision in a number of different chart types. Due to redundancy of the pair-wise comparisons, the tool also includes capabilities such a consistency check that describes the amount of judgement errors that are made during the prioritisation.

3.2 Operation

The participating anonymous organisation was given the task to use PARSEQ to reflect on a set of decisions made during prior releases. The case study was executed during a one-day session, with approximately 5 hours of efficient work.

Requirements sampling. A release that was launched 18 months ago was selected as reference release, and since then another release has been launched and yet another one is planned to be released in the near future.

The requirements database contains more than 1000 requirements that are issued before the reference release and implemented in either that release or postponed to one of the following ones. Of these requirements, 45 was considered a reasonable amount to extract. The requirements were equally allocated over the three releases: A, B and C, i.e. 15 were implemented in the reference release A, 15 in release B and another 15 were planned for release C.

Note that the releases were not equally large in terms of number of requirements, i.e. the samples are not representative. The 15 requirements from release A were selected among 137 requirements, while the releases B and C only consisted of 28 and 26 requirements, respectively as shown in Fig. 3.

The requirements were selected randomly from a range where the ones estimated as having a very high, or very low, development effort had been removed, since they are not considered as representative. Very similar requirements had also been excluded to get an as broad sample as possible, as well as very new ones as development costs had not been estimated.

All market changes, architectural decisions and new knowledge gained during the 18 months between the reference release A and release C could be applied. The selected requirements are all in the states Done or Accepted for implementation; no rejected or postponed requirements were considered in the analysis. The requirements sampling took approximately one hour and was performed by a developer before the session.

Re-estimation of cost and value. The re-estimation was performed to find out what requirements the organisation would have selected for release A if they knew then what they know now. With the knowledge gained since the reference release was planned, it is possible that a different set of requirements would have been selected. However, it is important to note that one additional requirement in the release would imply that another one has to be removed, in order to keep the budget and deadline. Furthermore, some requirements that were not implemented in release A were not elicited until release B or C, but if they had, it would have been good to include them.

The following question was used in the pairwise comparison of the candidates to the reference release: "Which of the requirements would, from a market perspective, have been the best choice for release A?". This question was carefully chosen with the objective of enforcing focus on the retrospective nature of the estimation. Thus, the assessment concerned the market value given what is known today, and not whether the decisions made during the reference release were correct or not given the knowledge available at that time.

The 45 requirements were re-estimated by using the Focal Point tool and pair-wise comparisons to prioritise them based on the selected question. The prioritisation

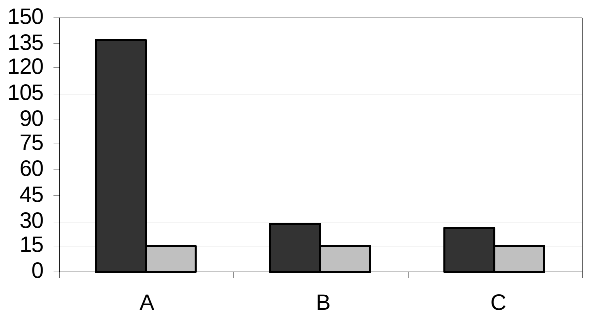

Fig. 3. Number of implemented requirements (dark grey) in each release compared to the sample (light grey).

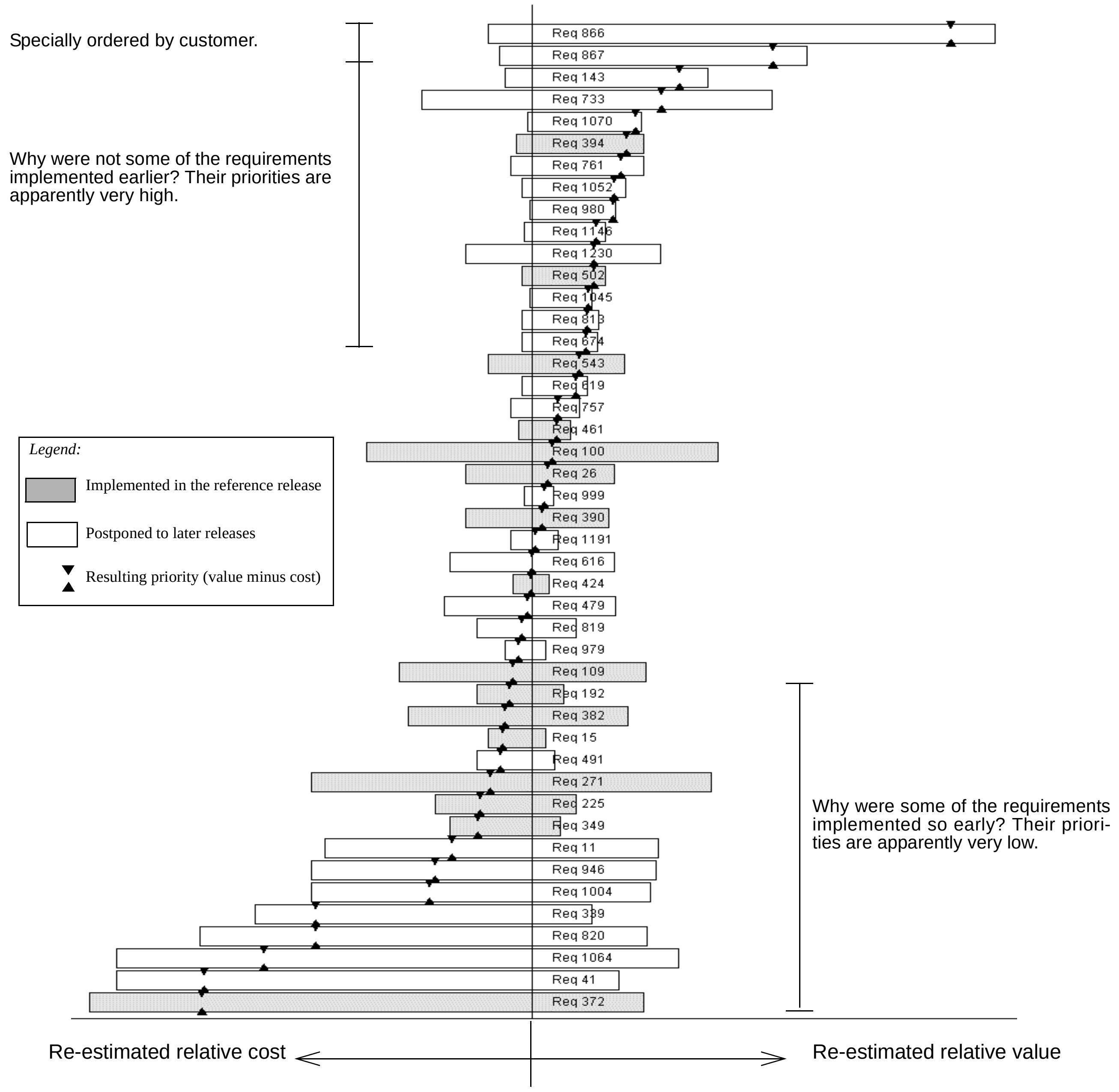

Fig. 4. Bar chart from the post-release analysis of the requirements in the database using the Focal Point tool.

was performed by a marketing person, who has good knowledge of customer demands, guided by a developer, and was attended by the two researchers. When uncertainties or disagreements of a comparison were discovered, the issue was shortly discussed to come to an agreement. The consistency check showed that the prioritisation was carefully performed and only two comparisons had to be revised and changed.

The total time of the prioritisation was just over one hour, in which 70 comparisons were made. The short time is thanks to the algorithms in the tool, which reduces the number of comparisons and points out the inconsistencies among the comparisons. Otherwise, the number of comparisons would have been n(n-1)/2, which in this case equals 990.

The development cost of the requirements that were actually implemented was known, while the development cost of the requirements that are planned for a coming release had to be re-estimated. However, it was decided to use the available cost estimations, since the estimates recently had been reviewed and updated.

A bar chart was created in the Focal Point tool to visualise and facilitate analysis of the decisions, see Fig. 4. The grey bars illustrate the requirements implemented in release A, and the white bars represent requirements implemented or planned for release B or C. The prioritisations are performed on a ratio scale and normalised to a relative value in the range between 0 and 1. Thus, it is possible to subtract the cost from the value, getting a *resulting priority*, which is marked by the black arrows in the bar chart [4]. The bars are sorted on their resulting priority from top down. Thus the bar chart shows the ideal order in which requirements should be implemented if only customer value and development costs were to be considered.

Some of the requirements were not identified in release A, but turned out to be important when they later were identified. Furthermore, requirements interdependencies, release themes and architectural choices complicate the situation and thus this ideal order is not the most suitable in reality.

In an ideal case, the requirements at the top of the bar chart would have consisted of requirements from release A. The requirements at the top of the bar chart are estimated as having the highest value and the lowest cost and should therefore be implemented in an as early release as possible. The requirements at the bottom are estimated as having the lowest value and the highest cost and should therefore be implemented in a later release or, in some cases, not at all.

The bar chart illustrates the discrepancies between the two estimation occasions and points out the requirements to discuss.

Root cause analysis. The bar chart is used in the Root cause analysis, to find out the rationale for the release-planning decisions. The discussion was attended by three representatives from the organisation: one marketing person and two developers, as well as the two researchers.

The top 15 requirements were scanned to find the ones that were estimated differently in the re-estimation, i.e. the ones that originate from release B or C. These were discussed to answer the main question “Why wasn’t this implemented earlier?” and motivations to the decision was stated by the participants. In a similar manner, the 15 requirements at the bottom of the bar chart were investigated, to find the ones that originate from release A and B. These requirements were discussed concerning the question “Why did we implement this so early?”. Notes were taken of the stated answers for later categorisation of the release-planning decision root causes.

After the meeting, the researchers classified the stated decision root causes into a total of 19 different categories, inspired by the notes from the meeting. A sheet with the requirements that had been discussed during the root cause analysis was compiled, which the organisation representatives used to classify the requirements. The result from the classification is displayed in Table 1 and Table 2, where 4 categories have been removed as they were not used.

Elicitation of improvements. Another purpose of the case study was to capture improvement proposals by encouraging the participants to, in connection with each requirement, state some weak areas in need of improvement. This also appeared to be difficult since each decision was dependent on the specific context or situation. Therefore, no list of improvement proposals was compiled at this stage. Instead, more generic improvement proposal areas were elicited by investigating Table 1 and Table 2 and the notes taken from the root cause analysis discussion. This is described below.

3.3 Results

The case study showed that it was possible to use the proposed method in practice. The release-planning decisions that were made in prior releases could be categorised and analysed and process improvement areas could be identified. The results indicate that the organisation has gained a lot of knowledge since the planning of the reference release, which is a promising sign of evolution and progress.

The causes for implementing requirements earlier than necessary are shown in Table 1. Most of the root causes originate from wishing to satisfy customer demands, either one specific customer or the whole market. However, the evaluation showed that the customer value was not as high as expected. On the other hand, it is difficult to measure “good-will” in terms of money, and therefore these decisions may not be essentially wrong. Other root causes of implementing requirements earlier than necessary concern implementation issues, such as incorrect effort estimations, which lead us to believe that estimations ought to be more firmly grounded. Another reason concerns release themes which is a kind of requirements interdependency that is necessary to respect. Developing and releasing small increments of requirements, in order for customers to give feedback early, is a good way of finding out more exactly what customers want, while assigning a low development effort.

As Table 2 shows, the reasons for implementing requirements later than optimal mainly apply to implementation issues. The category complying with the most requirements regards partial implementation in a first increment, which means that it was implemented earlier, but only partially and therefore the requirement remains.

The root cause tables and the material from the discussion were used in the investigation of possible improvement areas. Four areas were found which could be linked to the root causes and described below.

Trim the division of large requirements into smaller increments. The manner in which large requirements are divided into smaller increments can be more thoroughly investigated. The division can be done for several reasons: to get customer feedback at an early stage, to investigate alternative design solutions or to make small incremental improvements of the functionality. Root causes number 3

Table 1. “Why was this requirement implemented so early?”

	Root Causes	Req 192	Req 382	Req 15	Req 271	Req 225	Req 349	Req 372	Req 41
Implem. issues	RC1: Under-estimation of development effort	■	■		■				
	RC2: Part of release theme		■		■			■	■
	RC3: A quick fix to provide customers opportunity to give feedback		■		■				■
Customer issues	RC4: Requirement ordered by a specific customer								■
	RC5: Requirement specifically important for a key customer				■			■	■
	RC6: Over-estimation of customer value			■		■	■	■	■
	RC7: Impressive on a demo					■		■	■
	RC8: Competitors have it, therefore we must also have it							■	
	RC9: Competitors do not have it; gives competitive advantage			■		■			■

Table 2. “Why was this requirement not implemented earlier?”

	Root Causes	Req 143	Req 733	Req 1070	Req 761	Req 1052	Req 980	Req 1146	Req 1045	Req 813	Req 674	Req 866	Req 867
Implementation issues	RC10: Over-estimation of development effort	■							■	■			
	RC11: Insufficient understanding of scale-up effects			■				■					
	RC12: No good design solution available		■		■								
	RC13: Sub-optimal decision based on requirements partitioning		■										
	RC14: Only partial implementation in a first increment				■	■	■				■		
Cus. issues	RC15: Requirement ordered by a specific customer											■	■

and 14 deal with requirements developed in increments and the discussions resulted in the idea that the organisation would benefit from an improved increment planning.

Enhance the holistic view of related requirements. Some requirements were acknowledged as being related to other requirements due to involving the same feature. These would probably have benefited from creating a more holistic view of the release so that all aspects of the specific feature were accounted for. In some cases a feature involved several requirements and after implementing some of them the developers felt content. The related requirements could instead have been designed concurrently in one larger action to avoid a range of smaller ones. It would also have helped in identifying the most important requirements for that feature. These requirements relations could be taken into consideration more carefully as root cause number 13 describes.

Additional elicitation effort for usability requirements. It was recognised that the requirements dealing with the user interface did not fulfil some special customer needs, as described by root cause number 11. The problem concerned scale-up effects and could have been discovered through a more thorough requirements elicitation. Actions to take include building prototypes and asking customers with special user interface needs.

Improve estimations of market-value of features in competing products. It seems as many requirements were implemented with the objective of outperforming competitors, as reflected in root cause number 7, 8 and 9. However, looking too much at what competitors have or what may look nice on a prototype or demo may bring less value to the product than expected. The value estimations of the competitors’ products may need to be improved.

Improve estimations of development effort. Root causes number 1 and 10 concern over- and underestimations of the development effort. Results from an earlier study indicate that the release plan is very dependent on accurate time estimates, since the estimates affect how many of the requirements that are selected [8]. Under-estimation may result in an exceeded deadline and over-estimation may exclude valuable requirements. Improving this area may enhance release-planning and requirements selection quality.

4 Discussion

The case study participants found the one-day exercise interesting and instructive. They all agreed that it was valuable to reassess previous releases and reflect on the decisions made. It was during the root cause analysis that most learnings were made since the discussions between the participants were very fruitful. A set of improvement issues to bear in mind during requirements selection was assessed as valuable for future releases.

Despite the fact that 20 out of 45 requirements were assessed as belonging to the wrong release, there were few decisions that were essentially wrong. Keeping in mind the knowledge available at the time of the reference release, most release-planning decisions were correct, i.e. market opportunities and risks have to be taken, incremental development is applied and only a limited amount of time can be assigned to requirements elicitation and evaluation. However, no matter how successful organisation or product, there are always room for improvements.

There are a number of validity issues to consider in the case study. First of all, the data was not extracted from a representative sample because the releases varied in size. Therefore there are probably many more requirements from the largest release that would be interesting to consider. Since the data only included requirements that were implemented or postponed and no rejected requirements, there would be more decisions to consider in a more thorough evaluation.

The criterion that was used to capture the true value of the requirements appeared to be somewhat difficult to use. Since the development cost was known in most cases, it was difficult for the participants to concentrate on the customer value only, without implicitly taking the cost into account. It was also difficult to, in retrospect, consider the reference release and the value at that particular time without regard of the situation today.

The prioritisation itself is also a source of uncertainty; when not performed thoroughly, the bar chart may not show the appropriate requirements priorities. Nevertheless, the consistency check proved that the prioritisation was performed carefully and few judgment errors were made [6, 7].

Finally, the decision categories that emerged during the root cause analysis may not reflect the typical kinds of decisions. A different set of requirements would probably generate a different set of categories, and therefore these shall not be used by themselves. It is also possible that the categories are formulated vaguely or incorrectly, so that their interpretations differ.

The presented improvement areas are specific to the particular case study organisation and need to be examined in further detail to point out the exact measures to take. However, the participants state that the exercise itself, imposing thought and reflection, may be more fruitful than the particular improvement proposals.

5 Conclusions

The presented method for post-release analysis of requirements selection quality, called PARSEQ, was tested in a case study where candidate requirements for a previous release were evaluated in retrospect. The case study demonstrated the feasibility of the method in the context of the specific case and the results from the case study encourage further studies of the method. This may support the hypothesis that the method is generally applicable in the improvement of industrial processes for market-driven requirements engineering in product software development.

The following areas are interesting in further investigations of PARSEQ:

- *Include rejected requirements*. The case study only included requirements that were planned for implementation in the reference release or postponed to coming releases. It would be interested to go through the set of rejected requirements and see if there exist suspected inappropriate rejections, which may be of valuable input to the elicitation of improvements.
- *Selection quality metrics*. Given that the requirements sample is representative to the distribution of appropriate and inappropriate decisions, it may be possible to use PARSEQ to provide numerical estimations of the selection quality in terms of fractions of “good” and “bad” decisions.
- *Generalisation of root cause categories*. If many case studies applying PARSEQ are carried out in various contexts, it may be possible to derive a complete and generally applicable set of root cause categories that are common reasons for inappropriate decisions. This knowledge may be very valuable in the research of requirements engineering methods in the product software domain.

Acknowledgements

The authors would like to thank the participating anonymous organisation for the industrial requirements engineering expertise and confidential data, without which this study would not have been possible. We would also like to thank Magnus Höglund at Focal Point for contributing to this work with his valuable time and knowledge.

References

[1] Carlshamre, P., Regnell, B., “Requirements Lifecycle Management and Release Planning in Market-Driven Requirements Engineering Processes”, *IEEE International. Workshop on the Requirements Engineering Process* (REP’2000), Greenwich, UK, September 2000.

[2] Carshamre, P., Sandahl, K., Lindvall, M., Regnell, B., Natt och Dag, J., “An Industrial Survey of Requirements Interdependencies in Software Release Planning”, *IEEE International Conference on Requirements Engineering* (RE’01), pp. 84-91, 2001.

[3] Cleland, D.I., *Project Management,* McGraw-Hill, 1995.

[4] Fenton, N.E., *Software Metrics - A Rigorous Approach*, Chapman & Hall, 1994.

[5] Higgins, S. A., de Laat, M., Gieles, P. M. C., Guerts, E. M., “Managing Product Requirements for Medical IT Products”, *IEEE International Conference on Requirements Engineering* (RE’02), pp. 341-349, 2002.

[6] Karlsson, J., Ryan, K., “A Cost-Value Approach for Prioritizing Requirements”, *IEEE Software*, Sept/Oct 1997, pp. 67-74.

[7] Karlsson, J., Wohlin, C., Regnell, B. “An Evaluation of Methods for Prioritizing Software Requirements”, *Information and Software Technology,* Vol 39(14-15): 939-947, 1998.

[8] Karlsson, L., Dahlstedt, Å.G., Natt och Dag, J., Regnell, B., Persson, A., “Challenges in Market-Driven Requirements Engineering - an Industrial Interview Study”, *International Workshop on Requirements Engineering: Foundations of Software Quality* (REFSQ’02), Essen, Germany, September 2002.

[9] Lauesen, S., Vinter, O., “Preventing Requirements Defects: An Experiment in Process Improvement”, *Requirements Engineering* Vol 6:37-50, 2001.

[10] Paulk, M. C., Weber, C. V., Curtis, B., *The Capability Maturity Model: Guidelines for Improving the Software Process*, Addison Wesley, 1995.

[11] Potts, C., “Invented Requirements and Imagined Customers: Requirements Engineering for Off-the-Shelf Software”, *Proceedings of the Second IEEE International Symposium on Requirements Engineering* (RE’95), pp. 128-30, 1995.

[12] Regnell, B., Beremark, P., Eklundh, O., “A Market-Driven Requirements Engineering Process - Results from an Industrial Process Improvement Programme”, *Requirements Engineering,* 3:121-129, 1998.

[13] Robson, C., *Real World Research,* Blackwell, 2002.

[14] Sawyer, P., “Packaged Software: Challenges for RE”, *Proc. 6th Int. Workshop on Requirements Engineering: Foundations of Software Quality* (REFSQ’00), Stockholm, Sweden, pp 137-142, June 2000.

[15] Ulrich K.T., Eppinger, S.D., *Product Design and Development,* McGraw-Hill, 2000.

[16] Yeh, A., “Requirements Engineering Support Technique (REQUEST): A Market Driven Requirements Management Process”, *IEEE Second Symposium of Quality Software Development Tools*, pp. 211-223, New Orleans USA, May 1992.