Do Stakeholders Understand Feature Descriptions? A Live Experiment.

Rumyana Proynova and Barbara Paech

Software Engineering group, Institute for Computer Science, University of Heidelberg, Germany
{proynova, paech}@informatik.uni-heidelberg.de

Abstract. [Context and motivation] Requirements engineers need feedback from stakeholders on planned system features. The simplest way is to present feature descriptions to the stakeholders and ask for their opinion. [Problem/question] The feedback is only valid if the stakeholders' conception of the features represents the actual features reasonably well. Due to the highly abstract nature of software, it is possible that there is a mismatch between the stakeholders' idea of the feature description and the actual feature the software engineers intend to implement. [Method/results] We conducted a live experiment during the RefsQ 2012 conference. We used a questionnaire to measure the mismatch between the participants' understanding and liking of a list of software features and the implementation of these features, shown as a screencast of a system prototype. We found a correlation between the degree of understanding of a feature and the liking of that feature. There were no significant differences between features presented in different formats. [Contribution] This experiment shows first insights into the factors which contribute to a stakeholder's understanding of a feature description, and to his/her satisfaction with a software which contains these features.

1 Introduction

It is not feasible to involve end users in requirements elicitation in all projects, even though this could lead to higher quality requirements. Factors like the unavailability of end users (for example in global software development projects or off-the-shelf software products with no designated end users) or limited budget and resources, as well as company culture, can dictate that the requirements for the software are derived from other sources. In order to ensure that these requirements are aligned with the needs of the end users, the requirements engineers can let the users validate the requirements.

The constraints which preclude resource-intensive elicitation techniques are likely to also preclude similarly resource-intensive validation techniques. A technique, which produces adequate results but requires a comparably low level of effort, can enable early validation in projects where currently end users are not involved until the very late stages of the project such as testing or even roll-out of a completed product. Our research focusses on defining such a technique. The experiment described in this article is part of the research needed for the creation of this technique.

The technique is based on the well-known marketing concept of using a questionnaire to measure customers' expected satisfaction with product features [11]. Such a questionnaire lists a number of features and the prospective customer indicates his or her expected satisfaction on a Likert scale. While marketing literature provides research on using this technique, it cannot be used for software products without substantial extensions. The abstract nature of software and the often intricate dependencies between features make it difficult for the questionnaire respondent to give meaningful ratings for the individual features.

While the questionnaire technique has some distinct advantages, such as making efficient use of the time of the requirements professional (as there are no separate interviews with each user) and delivering quantitative data, it also has its drawbacks. The communication between end users and requirements engineers is very limited, and there is no convenient channel for inquiries, clarifications and discussion. All information a user gets about a feature is its description. If users misunderstand a feature description, they may approve a feature they do not need, or declare that they need a feature which is in fact useless. They may also have ideas about how to improve the features, but may not inform the requirements engineers, because they are unsure about the proper communication channel.

For the communication between the requirements engineer and the end user to function properly, it is important that the requirements engineer asks the right questions, the users understand them in the way they were intended, and answer truthfully. This article describes an experiment which we conducted in order to gain more insight into the first two points. Our experiment does not address the possibility of users intentionally giving wrong answers.

In the next section, we describe research related to our experiment. In section 3, we describe how we designed our questionnaire and how we conducted the experiment itself. Section 4 describes our hypotheses, our results, and some explorative conclusions we made in addition to the predefined hypotheses. The last section presents our intended future work on this topic.

2 Related Work

Our experiment uses a questionnaire for measuring the users' satisfaction with a product. There are several techniques in marketing for similar types of analysis. Prominent ones are concept testing [10], conjoint analysis [7], importance performance analysis [9] and SERVQUAL [1].

None of these methods can be used directly with software features. Concept testing presumes that a customer is able to make a buying decision based on an advertisement presented on a magazine page. This is useful for products like toothpaste, which only have a few simple features, but is not feasible for software products. Conjoint analysis requires a customer to make pairwise comparisons between each possible level of many factors, for example a product can have the factors *product warranty (levels: 1 year, 2 years)* and *support options (levels: telephone hotline, online ticket submission)* and the customer has to rank all four possible combinations. As the number of comparisons to be made grows exponentially with the number of factors present, it is not suitable for a products with multiple features. Importance-performance analysis needs easily quantifiable features, for example *fizziness* for a soda. SERVQUAL includes a hierarchical set of service qualities, which can not be mapped to a software product without changes. All four of these techniques assume that the name of a feature conveys enough information about it to be perfectly understandable for a customer, and the studies which use these methods are done on products with simple, well-known features where understandability is not a problem.

Our research is based on software features. Software features are an important part of how users understand software. They are often used in software specifications, especially in the context of software product lines. [2], [12]. Other types of specification can often be broken down into individual features, the way we do this in the experiment described here - we produce our features from a software specification written as user tasks [8] and user stories [3].

3 Experiment Design

The experiment was conducted during the empirical track of the RefsQ 2012 conference, with 56 conference visitors participating. Each participant received a questionnaire and was asked to answer the first part of it. This part contained software feature descriptions and questions about them. Then the participants were shown a screencast of a software application implementing the features from the first part, and had to answer questions about the feature implementation.

For the experiment, we created a software requirements specification and a software prototype implementing the specification. We chose to implement a personal finance software product for managing receipts. This choice had several advantages. First, this application could be made simple enough to cover the complete specification of functional requirements in the limited time available for the experiment. Second, this class of software product is not especially common, and we expected most of the participants to not have made experience with similar software products before. This reduced the chance of preconceptions formed through contact with similar software products to skew the results. Third, using this software does not require any special knowledge. Had we chosen a software supporting a process which is only performed in certain professional fields or hobbies, we would have had to control for the participants' experience in these fields or hobbies.

We used the features from our requirements specification for the features description in the questionnaire. We recorded a screencast of the first author using the software, and used this recording for the experiment. The questions in the questionnaire were derived from our research goals.

3.1 Research Goals

The purpose of this experiment was explorative research. We wanted to get first insights into how users understand feature descriptions and how a requirements engineer can create a questionnaire which reflects the true future satisfaction of an end user with given software features. Such an instrument can never be perfectly accurate, but we want to achieve a degree of accuracy high enough for it to be used in decision-making about which features should be implemented in a software product.

We had three main research goals. We wanted *to a*) understand to what degree an end user's satisfaction with an implemented feature correlates with a set of factors we assumed are connected to satisfaction, most importantly to what degree self-predicted satisfaction correlates with actual satisfaction; *to b*) to find further factors, beside the ones we assumed in the previous goal, which could possibly play a role in self-reported expected satisfaction; and *to c*) to test whether different feature presentation formats influence the understanding of features.

Correlation of end user satisfaction with proposed factors While marketing theory often works with complicated concepts to describe satisfaction, a questionnaire study has to use simple, universally understood concepts in a questionnaire, else it risks getting incomparable results due to a confusion of what is being asked. For this experiment, we chose one of the simplest concepts possible: liking. As we are interested not only in knowing self-reported liking, but also in how accurate the answers are, we also included questions about the understanding of what a feature is about. A misunderstood feature will lead to answers based on a false conception of the feature and therefore reduce the usefulness of a questionnaire. We asked about the liking and understanding both before and after the participants saw the feature implemented in a software demonstration. We formulated multiple hypotheses about the possible relation between how well participants believed they have understood a feature before and after implementation, and how much they liked it before and after implementation. Below is a list of these hypotheses, with a possible explanation for each hypothesis given in parentheses.

Hypothesis 1 Accurate understanding of an implemented feature is related to a feeling of understanding it before seeing it implemented. (Users know whether they have understood a feature).

Hypothesis 2 Liking of a feature before seeing it implemented is related to a feeling of understanding it before seeing it implemented. (Users like conceptions they understand).

Hypothesis 3 Liking of a feature after seeing it implemented is related to a feeling of understanding it before seeing it implemented. (Users like features which were clear from the beginning).

Hypothesis 4 The deviation in liking a feature before and after seeing it implemented is negatively related to a feeling of understanding it before seeing it implemented. (The better users have understood a feature, the better they can predict whether they will like it. Alternatively, the causation could be reversed: The more the users like a feature, the better they are able to understand it with a short description).

Finding further relevant factors Unlike the other research goals, this one was based on qualitative research. We collected feedback about what participants thought influenced their answers to the questionnaire. There were no preformulated hypotheses for this goal.

Different feature presentations Software requirements can be documented and represented in multiple ways. Some of these are not suitable for use in a questionnaire for non-technical users. We considered several formats which were more or less self-explanatory. As we were concerned with a clear understanding by a user who does not have the possibility to ask for clarification, we decided to compare formats which are more or less close to a user's point of view. We included user stories and user tasks in our questionnaire.

We based our user stories on the guidelines given by [3]. They represent the type of requirements documentation used in modern agile software development approaches such as Scrum. Each user story is a short description of what the user can do with the system, which is supposed to be documented on a separate card and used primarily for release planning and developer task specification. A user story is always written from the point of view of the end user, it is closed and supports a user achieving a goal. The granularity is determined by the development process: user stories do not contain details, so each of them must be small enough to be sufficiently described in one or two sentences, and to be implemented by a developer within a single sprint. Each user story should be independent of other user stories. For the survey, we use each user story as a separate feature. While a user story card can contain constraints (non-functional requirements) or acceptance tests, we do not include these in the survey, as they do not have a direct analogue in the other two approaches.

For user tasks, we use the guidelines proposed by [8], specifically the so-called "task and support" approach. In this approach, the subtasks are combined with proposed solutions which can be implemented in the system. User tasks are written close to a user's point of view. Each task is based on a goal the user wants to achieve. Subtasks do not focus on either user or system; they "describe what the user and the computer do together" on a domain level. They are written in the imperative mood, to hide who does what. The solutions may specify what the system does, but this is not always the case. User tasks do not provide an ordered sequence of steps the way use cases or scenarios do. However, they package small substeps into bigger units, the tasks. Each task has one user goal and should have closure. This means that a task provides more context information to the user than formats which record each subtask-sized requirement independently. For the purposes of our survey, we define each solution from the Task & Support approach as a product feature.

We formulated several hypotheses on the differences which formats can cause in the answers.

Hypothesis 5 The closer a format is to the user's point of view, the better the feeling of understanding it before seeing it implemented. (Users feel they understand feature descriptions better when they are described from their point of view).

Hypothesis 6 The closer a format to the user's point of view, the better the understanding after seeing it implemented. (Users understand features better when they are described from their point of view).

Hypothesis 7 The closer a format is to the user's point of view, the less deviation is there in liking the feature before and after seeing it implemented. (Users can better predict their liking of a feature when it is described from their point of view).

3.2 Features

The requirements specification used for the experiment contained fifteen features. The complete specification in the user task format is shown in section 7. We tried to make the specification as realistic as possible. We did not polish it to be the best specification we can produce, but included features which we assumed were not very good from the end users' point of view. This was done in order to have variability in the features' liking, as we feared that, if all users like all features uniformly, we would not be able to see any effects. We also included a "double" feature, where the users had to rate two either-or alternatives, none of which were very user-friendly. The reason was that we wanted to force the participants to dislike at least one feature, again to have more variability. Also, the fact that one feature excluded the other was unusual (there was no obvious technical reason why both could not be implemented at the same time), and we assumed that some participants would be misled to believe that both features will be included in the software allowing the user to choose between them at runtime. We hoped that this would create a situation where users believe that they have understood the feature before seeing the implementation, but recognize that their understanding was not accurate after seeing the screencast.

Beside variability in liking, we also wanted to create variability in understanding. For this, we knowingly kept some features ambiguous, omitting important details. We feel that this made the situation more realistic. First, the literature on user stories and user tasks suggests that these descriptions should not contain very detailed information in order to prevent overspecification [8], [3]. Second, in industry projects we have been involved with, we have often seen features described at the same or at a more abstract detail level as the one we used, but never at a lower, information-richer detail level. Thus, including intentionally ambiguous feature descriptions made the experiment situation more realistic.

We developed a GUI protoype based on our specification and created a realistic test data set including actual receipts. The prototype was created in Java and had only rudimentary functionality, but, when used with the test data set, it could create a convincing illusion of actually having the functionality described in the specification. We created screencasts of one experimenter using the features described in the specification, accompanied by an audio commentary explaining what is happening on the screen. These screencasts were shown to the participants during the experiment, as explained in the next section.

3.3 Questionnaire

The questionnaire consisted of four parts. The participants were given time to answer parts one and two, then asked to wait for a software demonstration.

Part one contained general demographic questions. Participants were asked about their expertise in requirements engineering and their experience with different requirements representation formats. The data from these answers is not evaluated for any of the hypotheses listed in the research goals; rather, it is used to describe relevant demographic factors of the experiment participants.

In part two, the participants were asked to read a requirements specification and answer questions about the features it contained. There were two types of questionnaires, which differed in this part. One type had user stories, each printed in a simple box without a number, representing a story card. The second type had two user tasks formatted as described in [8] with each solution containing a single feature. The wording of all features was equivalent in both formats except for rules prescribed by the format.

For each feature, participants were asked to answer two questions:

Question 1 "My conception of the way this feature will be implemented is ...", followed by a five-point Likert scale labelled clear/vague/non-existent (All Likert scales used in this questionnaire were five-point with only the first, third and fifth point labelled). This question was used to measure the feeling of understanding before seeing the implemented feature.

Question 2 "I think I will ... this feature when it is implemented", followed by a Likert scale labelled like/be indifferent/dislike. This question was used to measure liking before seeing the implementation.

These two questions were followed by a prioritization task. We have not yet completed the evaluations based on the data from the prioritization, so we do not report on it further in this article.

In part three, the participants again were shown each feature, with three new questions per feature. They were instructed to wait for a video demonstration of the features before answering the questions. Each demonstration contained two or three features, so that features which were solutions to the same task as defined in the user tasks were bundled in the same demonstration. Participants were given time to answer the questions about all features in a demonstration before the next demonstration was presented.

Question 3 "The feature corresponds ... to my previous conception", with a Likert scale labelled very well/somewhat/not at all. This question was used to measure accurate understanding.

Question 4 "The implemented feature differs from my previous conception in the following ways:", followed by empty space for a freely formulated answer. This question was used for generating ideas for other relevant factors (second research goal).

Question 5 "I ... the feature the way it is implemented now.", followed by a Likert scale labelled like/be indifferent/dislike. This question was used to measure liking of a feature after seeing the implementation.

Part four included one open-ended question, and a feedback section. In the question, we asked the participants to give us suggestions on how they would have improved the feature descriptions to improve their understandability. In the feedback section, we asked for freely-formulated feedback on any part of the experiment, the conduction, and the questionnaire. The answers from this part were used to generate new ideas for relevant factors (second research goal), as well as for recognizing possible threats to validity, and as information on how to structure a follow-up experiment.

4 Results

We used the statistical language R to evaluate the hypotheses stated in section 3.1. We could not confirm the hypotheses about the differences in the description formats, but found a good correlation in most hypotheses about the relations between the understanding and liking of features before and after seeing their implementation. The qualitative analysis of the open-ended questions as well as the feedback the participants gave during a discussion session at the conference led to interesting new ideas which we will explore in further experiments.

4.1 Correlation of End User Satisfaction with Proposed Factors

To this research goal, we calculated a correlation coefficient for hypotheses 1 through 4. As this is not a significance test for a parameter, we can not offer a significance level; rather, we can say that the higher the absolute value of the correlation coefficient, the more connection there is between the two phenomena. The results are listed in table 1. Figure 1 gives an overview of the correlations we found.

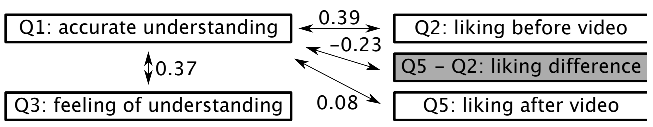
Q1: accurate understanding
0.39
Q2: liking before video
−0.23
Q5 – Q2: liking difference
0.37
Q3: feeling of understanding
0.08
Q5: liking after video

Fig. 1. Concepts measured and their correlations

We found that most hypotheses of this research goal, with the exception of hypothesis 3, were confirmed. The correlation coefficients were clearly different from zero (so the two factors are related), and the sign direction was as the

Hypothesis	Questions	Corre-lation
Accurate understanding of an implemented feature is related to a feeling of understanding it before seeing it implemented.	Q 1 and Q3	0.37
Liking of a feature before seeing it implemented is related to a feeling of understanding it before seeing it implemented.	Q 2 and Q 1	0.39
Liking of a feature after seeing it implemented is related to a feeling of understanding it before seeing it implemented.	Q 1 and Q5	0.08
The deviation in liking a feature before and after seeing it implemented is negatively related to a feeling of understanding it before seeing it implemented.	(Q 5 - Q 2) and Q1	-0.23

Table 1. Hypotheses about the connections between the concepts

hypothesis predicted. We conclude that a clear conception is an important factor in the participants' answers to a questionnaire with software feature descriptions. We propose the tentative interpretation that people tend to be unsure about what answer to give for a feature they can not picture clearly. We cannot claim that the data confirms this interpretation; we need more research, not based on pure correlation, to find out whether this interpretation is correct.

4.2 Finding Further Relevant Factors

We are interested to know what influences the answers the participants give to questions such as "Do you like this feature" and also their ability to envision how a feature will be implemented. The discussion and the answers to the open questions provided us with ideas about such factors.

Participant has encountered feature before. We chose an unknown software product, in order to avoid the situation that participants have preconceived notions about features based on their previous experience with such software. The discussion made us realize that this effect can emerge not only on the level of a software application, but also on the level of a single feature. The participants reported that they had a good understanding of features which they had previously encountered in unrelated software products, and that they had difficulty creating a clear concept of features specific to our software. The data confirms this claim: the features with the highest arithmetic mean in the answer to question 1 were the ones related to printing (feature 14, $\mu = 4.75$)[1], exporting a report (feature 15, $\mu = 4.44$) and saving data in a file (feature 7b, $\mu = 4.16$).

[1] As the results are measured on a five point Likert scale, the minimum possible score is 1, the maximum possible score is 5

This functionality is common in many types of software, and probably most of our participants have encountered it before.

Participant cannot imagine a good use for the feature. This was a comment left by one participant in the feedback part. He or she claimed that it was very hard to build a clear conception of the feature when he or she did not know why it could be useful or what they would want to use it for. This is an interesting comment which is in line with the recognized importance of rationale in requirements engineering [4]. In a follow-up experiment, we will include questions designed to measure the influence of this factor on the understanding.

Participant has no emotional attachment to the software product. This was suggested by a participant who claimed that he or she marked the "indifferent" option for all features not because of the relative merit of the feature as compared to other features in the description or to other possible features implementing a similar functionality, but because he or she found the software very "boring" and was indeed indifferent to anything which had to do with the software product we presented. This finding is in line with results of marketing research. First, modern theories of consumer satisfaction agree that satisfaction has both an emotional and a rational dimension. We chose to ask a single (expected) satisfaction question in this experiment, and formulated it using a word associated with emotions: *like*. It is possible that another formulation, such as one using a word associated with rational utility like *useful*, would have produced a different result at least from this user. However, marketing literature emphasizes that a customer should create an emotional connection to a product for satisfaction (as made popular by the AIDA principle - a customer's relation to a product starts with simple Attention, but it has to include Interest and Desire before they Act on it [5]), so maybe, while a rational formulation of the question could have produced different results, they would not necessarily have reflected the user's satisfaction well enough. This is an interesting connection we plan to research in more depth in future experiments.

Suggested improvements to feature presentation At the end of the questionnaire, we asked the participants to leave their assumed role of end users and reflect from the viewpoint of requirements specialists. We asked them to give us suggestions on how to improve the feature descriptions in order to achieve a better understanding and more accurate answers about the expected satisfaction. We received many suggestions, some of them given by more than one participant. We agree that all these suggestions have the potential to increase the understanding, but our background is a specific situation: a project with limited resources in an early phase of requirements elicitation and validation. Thus, we evaluated the suggestions on two criteria: the potential for increasing the users' understanding and the feasibility of a project team creating the suggested artefacts in this project phase. We will do further research on the suggestions which scored high in both dimensions.

Participants' suggestions for individual features While evaluating the answers to the open-ended questions, we observed that the participants often offered suggestions for improving a feature, or at least could articulate what they dislike in a feature. Without receiving any specific guidance, they often made remarks which could be clearly assigned to the desire for a different user interface or different functionality, and they made clear what they would like to see instead of our solution. This effect was especially strong in features which were intentionally designed to be user-unfriendly, but it also existed in other features. The users even noticed potential problems which we had not noticed in our specification, but which sounded like true problems (as compared to the desire to have something in a different way without being objectively better) and could be solved without much additional development effort (in the case that the application was actually implemented).

While we are aware that our participants were experienced software engineers and thus can be expected to come up with such suggestions at a higher rate than the general population, the suggestions they had were often simple, interaction-specific changes which are not beyond the reach of users without a background in software engineering. For example, a frequent suggestion was that the report should be exported in common office formats, such as Word of PDF. Other suggestions showed that the participants were evaluating the features from the point of view of a user and not of a software engineer, for example the frequent suggestion that the text recognition should function reliably enough that the user does not have to confirm and correct the results. While this is a very understandable desire of a user, we expect software engineers to consider technology limitations before writing such a suggestion. We found no indication that such limitations had been considered, except for one participant who did not suggest perfect recognition, but instead suggested that the software displays reliability numbers (how confident it is that a given recognized text string is correct). Thus, we expect users without experience in software engineering to be able to give similar feedback, even though we expect them to have a lower suggestion rate and a lower ratio of useful to infeasible feature change suggestions. This expectation has yet to be confirmed in future research before it is incorporated in our technique.

4.3 Different Feature Presentations

One of our research goals was to compare the liking and understanding between features described in different formats. The used formats were rather similar in their degree of closeness to the users' point of view, but differed in the amount of context offered, with user tasks bundling related features together and containing additional information per task, while the user stories were represented on unconnected "cards". Thus we expected to see differences between the two.

We did two comparisons in parallel. The first one was a straightforward t-test of the arithmetic mean calculated separately for both types of feature description. The second one was an ANOVA test, which, with only one pair of treatments, resulted in a simple t-test for the variance of both distributions [6]. In the following, we call the participants who answered the user task questionnaire *user task group* and the participants who answered the user story questionnaire *user story group*.

Table 2 shows the exact formulation of our hypotheses. The column *Question* contains the number of the question which was used for evaluating the hypothesis. We evaluated by comparing the answers to this question given by the two groups. The last hypothesis included building a difference of the answers of two questions and comparing the mean and the variance of the difference. None of the hypotheses could be confirmed at a significance level of 5 %, neither when comparing the means nor when comparing the variances.

Hypothesis	Question	Result
The closer a format is to the user's point of view, the better the feeling of understanding it before seeing it implemented.	Q 1	not confirmed
The closer a format to the user's point of view, the better the understanding after seeing it implemented.	Q 3	not confirmed
The closer a format is to the user's point of view, the less deviation is there in liking the feature before and after seeing it implemented.	Q 2 - Q5	not confirmed

Table 2. Hypotheses about different feature presentations

We could not confirm any of the hypotheses we created for this research goal. Our conclusion is that the format in which feature descriptions are represented does not matter for the understanding or the liking of features, at least when we compare user tasks to user stories. The richer context of user tasks does not seem to lead to a better understanding. In the future, we plan to research the same hypotheses, but to include a format which is far removed from the users' point of view, for example sentence templates. The templates described e.g. in [13] begin with "The system shall" and continue to describe what the system does, instead of what the user does using the system.

4.4 Threats to validity

Conclusion validity We identified two possible threats to conclusion validity. First, we are not aware of any objective measures for the variables we measured, so we had to rely on self-reporting. Second, we ran a large number of hypotheses on the same data set. This is problematic for confirming a theory, as it lowers the actual significance level, but we used significance tests only for exploratory research and do not claim that the results are conclusive.

Internal validity The quality of the instruments we used for our experiment can also have compromised our experiment validity. First, users may have misunderstood questions due to ambiguity. We tried to counter this by presenting the questionnaires to coworkers not involved in the project and asking them whether they understood the questions the way we intended them. Second, we used screencasts of the software. The users' true judgement of features may have been incomplete, because they did not work with the software themselves.

Construct validity The questions we asked may not be best suited to measure the concepts of understanding and liking. For example, some users may have found it hard to answer whether their understanding had been accurate, as they might have had difficulty remembering how they imagined the features before the screencast.

External validity The most serious problem with our experiment is that the participants were software engineering experts and not typical users with non-technical background. The situation we used in the experiment was also not perfectly realistic. The users did not know why they should use the software. The feature descriptions may also not represent user stories well enough. We created them as tasks first and "translated" them into user stories. It is possible that requirements written after user story principles would have had a different content or granularity.

5 Conclusions and Future Work

Our experiment is based on the assumption of a certain project situation - the need to validate features with end users who were not involved in the feature elicitation, under limited resources. We propose that in this case, a questionnaire can be used for the validation, and the experiment aimed at finding out how accurate such a questionnaire can predict the users' future satisfaction with the software. We had three main research questions. Two of them were based on assumptions we made, and tried to confirm a number of hypotheses related to each assumption.

We could confirm almost all hypotheses related to the assumption that the ability to build a clear conception of a future implementation based on just a feature description has a strong influence on the answers to an expected satisfaction questionnaire. We plan to continue research focussed on this understanding and ways to improve it, in order to reduce the undesirable influence of low understanding on the rating of features on a scale for expected satisfaction.

We could not confirm the hypotheses related to the assumption that the feature description format has an influence on the answers to an expected satisfaction questionnaire. This finding has important consequences for our proposed technique: if every format is understood well enough, the technique can be used in projects employing any format without the need to translate the requirements to a specific format suited to the technique.

The open-ended questions as well as the feedback section allowed us to find other possible factors influencing the accuracy of self-reported satisfaction expectation. We intend to measure the influence of these factors in future research.

The future research questions mentioned above will be addressed in a future experiment, conducted with university students without a software engineering background. The design of the new experiment will be altered in order to measure the effects we did not measure in this experiment, especially the ones we found suggested in the open-ended questions and the written and verbal feedback. It will also contain the old questions, so it will provide confirmation for the findings already presented here. The use of a different demographic composition will address some of the threats to validity discussed in the previous section.

6 Acknowledgements

We thank the RefsQ organizers and especially the empirical track chair for giving us the possibility to conduct this experiment. We also thank all RefsQ participants who participated in our experiment.

The experiment is part of the project VaREMed (Value based requirements for medical software), funded by the Deutsche Forschungsgesellschaft.

References

1. Buttle, F., SERVQUAL: review, critique, research agenda, European Journal of Marketing, vol. 30, issue 1, MCB UP1996
2. Clements, P. and Northrop, L., Software product lines, 2001, Addison-Wesley
3. Cohn, M., User stories applied: For agile software development, 2004, Addison-Wesley
4. Dutoit, A.H. and McCall, R. and Mistrík, I. and Paech, B., Rationale management in software engineering, 2006, Springer
5. Ferrell, O.C. and Hartline, M, Marketing strategy, 2005, Thomson South-Western
6. Freedman, D. and Pisani, R. and Purves, R., Statistics, 2007, W.W. Norton & Co
7. Giesen, J., Volker, A., Requirements interdependencies and stakeholder preferences, Proceedings of the IEEE joint international conference on requirements engineering, IEEE 2002
8. Lauesen, S., User interface design: a software engineering perspective, 2005, Addison-Wesley
9. Martilla, J. and James, J., Importance-performance analysis The journal of marketing, JSTOR 1977
10. Moore, W., Concept testing, Journal of business research, vol.10, issue 3, Elsevier 1982
11. Oliver, R.L., Satisfaction: A behavioral perspective on the consumer, 2010, ME Sharpe Inc
12. Ruhe, G. and Saliu, M.O., The art and science of software release planning, IEEE software vol. 22, issue 6, 2005
13. Rupp, C., Requirements Engineering und -Management, 2009, Hanser

7 Appendix: Feature descriptions as user tasks

Task 1	Digitize receipt
Start: **End:**	The user has received one or more receipts The data from the receipts are archived
Subtasks	**Solution**
Import receipt	1 The system imports a picture of the receipt. 2 The system makes it easy to prepare the picture for recognition. 3 The system recognizes the text in the picture. 4 The system guesses a single tag and applies it to each expense item. The user doesn't predefine any tags.
Check and correct receipt data	5 The system allows the user to change any part of the recognized content in a receipt. 6 The system allows the user to change the tag for each expense item separately.
Archive data	7a) Option 1: The system archives the data on the servers of the system vendors (cloud storage). No local saving is possible. All data is saved automatically. 7b) Option 2: Instead of cloud storage, the system archives the data locally. The user has to trigger the saving. 8 The system offers export of the receipt data.

Task 2	View expenses report
Start: **End:**	The user needs information on expenses The relevant information has been viewed and possibly printed.
Subtasks	**Solution**
Select the input for the report	9 The system offers the user to select the receipts to be used in the report. 10 The system offers several search and filter options for finding relevant receipts in the receipt list. 11 The system offers templates for different types of report. 12 The system allows the user to input parameters for the report, e.g. a month for a report which shows expenses for a given month.
Produce report	13 The system processes the data needed for the report and shows onscreen a report in a print-friendly layout. 14 The system allows the user to print the report. 15 The system allows export of the report data.