Feature Prioritization Based on Mock-Purchase: A Mobile Case Study

Alexander-Derek Rein[1] and Jürgen Münch[2]

[1] Technical University of Munich, 85748 Garching, Germany,
alexander-derek.rein@in.tum.de
[2] University of Helsinki, Software Systems Engineering Research Group,
00014 Helsinki, Finland,
juergen.muench@cs.helsinki.fi,
WWW home page: http://www.sserg.org

Abstract. As development teams' resources are limited, selecting the right features is of utmost importance. Often, features are considered right if they result in increased business value at acceptable implementation cost. Predicting implementation cost and prioritizing features is well documented in literature. However, there has only been little work on the prediction of business value. This article presents an approach for feature proioritization that is based on mock-purchases. Considering several limitations, the approach allows key stakeholders to depict the real business value of a feature without having to implement it. Hence, the approach allows feature prioritization based on facts rather than on predictions. The approach was evaluated with a smartphone application. The business value of two features which were subjectively considered to be equally important was investigated. Moreover, the users were assigned different price categories for the features. Combined with live customer feedback, the approach allows us to identify an adequate pricing for the features. The study yielded insightful results as it showed which of the features incorporates higher revenue as well as how users react to the approach. It contributes to the body of knowledge in requirements engineering and software engineering as it enables practitioners to select features based on facts rather than predictions and to find ideal price points.

Key words: feature prioritization, requirements prioritization, live customer feedback

1 Introduction

As practitioners as well as academics agree upon, feature prioritization is crucial for software development as it helps to deliver value to customers sooner. Specifically small development teams need to make sound judgements about the features they develop as they are limited to developing a narrow set of features. When investigating requirements prioritization in practice, Bakalova et al. [1] found that the key prioritization criteria is business value, e.g., revenue. Further

important criteria are any risks and dependencies associated with the features.

The most common prioritization techniques such as the Kano Model of Customer Satisfaction or the Relative Weighing Model are mainly based on individual stakeholders' assessments on the business value of features.

However, it is undisputed that stakeholders' predications are affected by contextual biases[1]. Empirical work has shown that such biases can affect the estimates significantly [2]. On top of that, decision-making literature reports how a set of general biases can affect judgments. Such biases have shown to apply to the software domain as well [3].

Several approaches seek to address these issues by leveraging end-user feedback. Usually, end-users are surveyed in order to find out which features entice users to purchase, cause users to upgrade, or increase the business value in a different way. However, such approaches assume that end-users have clearly defined product priorities. Hence, these methods suffer from contextual biases as well. The reasons why end-users' feedback suffers from a lack of liability are manifold:

- A user who answers the question Would you buy this feature? in the affirmative does not necessarily really purchase the feature when it is available.
- Most users do not have clearly defined product priorities. Hence, they might say "I want them all" when presented with a set of features [4].
- Users might favour a particular feature over other features and therefore downgrade the other features on purpose.

As a step towards overcoming such deficiencies, we present an approach which aims at eliminating such biases. The key idea behind the approach is to have users mock-purchase features which are actually not yet implemented. Mock-purchase means that the purchase processes of the users' smartphones are emulated to resemble an organic purchase process. In the context of a smartphone application, this means emulating the In-App Purchase (IAP)[2] construct. Hence, users consider the features as actual IAPs prior to mock-purchasing them. As a result, there is no contextual bias. After the purchase, users are made aware about the methodology and informed that they were not charged any money. Moreover, their feelings towards the approach are surveyed, e.g. with the question *Are you okay with the approach?*. Personal feedback can be provided as well.

If users show interest in a feature, e.g., by loading the feature's price but do not purchase the feature, they are surveyed as well. Thereby, the approach allows us to find out whether the feature's pricing is too high.

In order to evaluate the approach, it was implemented in a smartphone application. Prior to that, two features which were considered equally important by the app author were derived from feature requests users had made.

[1] *Contextual Bias* occurs when a decision is influenced by specific knowledge about the case and applies this knowledge in order to influence the prioritization outcome.

[2] In-App Purchases are features or add-ons to a smartphone application that can be bought with real money. In-App Purchases are very convenient to use as smartphone users can pay for the features simply by pressing a button as their payment information is already stored on their phone.

The case study is structured as follows. Section 2 reviews related work, Section 3 describes the research method, Section 4 highlights study execution details, Section 5 presents the study results, Section 6 discusses validity issues, and finally Section 7 concludes and introduces directions for future work.

2 Related Work

Feature prioritization is an interdisciplinary problem. Therefore, there exist many approaches in other disciplines such as market research which might also be relevant for software engineering.

Several feature prioritization approaches are based on the Kano Model of Customer Satisfaction (Kano). Kano is a questionnaire-based approach that was originally developed in order to investigate customer satisfaction [5]. However, it is also commonly used in order to prioritize features in the context of requirements engineering [4]. As such, it surveys end-users with both functional and dysfunctional questions. The results can be assigned to a ranking scheme which allows to distinguish between essential and differentiating attributes. The question *How would you feel if this smartphone had GPS capabilities?* is an example of a functional question whereas *How would you feel if this smartphone would not have GPS capabilities?* is an example of a dysfunctional question. However, such questions provoke contextual bias, as only few users would dislike having GPS capabilities on their phones. As a result, it is hard to differentiate between features.

As feature prioritization is a cross-discipline problem, there have been efforts to solve the problem outside the software engineering world as well. Hohmann's Innovation Games are popular in primary market research. The game Buy a Feature as an example seeks to eliminate contextual biases by assigning a price to each feature. Users are then provided with play money to buy the features. However, Buy a Feature does not eliminate contextual biases, either, as users pay with play money rather than real money. As a result, users are more willing to spend money and are very likely to spend all their play money whereas in real conditions many users do not want to pay at all and will most certainly not spend all their money.

Carrenño and Winbladh suggest *ASUM*, an approach that allows us to automatically derive feature requests from mobile application comments. Their approach is helpful as it decreases the time required to extract features from the topics as well as finding out how often those were requested [6].

An approach that seeks to prioritize features not only based on user input but also on stakeholder judgements is Karl Wiegers Relative Weighting [7]. The approach allows finding out about which features to implement and what priority order. In Relative Weighting, weights are assigned to the relative benefits, relative penalties, relative costs, and relative risks of features by the developers and product owners. Subsequently, users are asked to weight the features. In fact, assessing benefits and penalties is similar to functional and dysfunctional questions in Kano. Based on the resulting numbers the priority per feature can

be calculated.

Besides, finding out about a feature's priority, Relative Weighting provides ground for discussions as benefits, penalties, costs, and risks assigned to features were objectively assigned.

3 Research Method

The research method is based on the guidelines for conducting case studies by Runeson et al [8].

3.1 Rationale & Objective of the Study

The rationale behind the case study is to find out whether the approach can contribute to the body of knowledge by eliminating contextual bias and thereby enabling feature selection based on facts rather than predictions. Moreover, whether the approach allows us to find out about how to set the price of features is under investigation.

The objective of the study design is threefold:

- Eliminate contextual bias by having users mock-purchase features.
- Randomly assign users different prices for the features in order to find out adequate price points. At the same time, collect information on how the users react to the pricing in order not to resent users.
- Collect information on why generally interested users decide not to purchase the feature (e.g., too expensive) in order to gain more insight.

As not all features qualify for being implemented as mock-purchases, such features need to be identified in the product backlog and treated with other methods.

3.2 Study Context

The context of the case study is a smartphone application called Track My Life (TML) which is developed by the study author. TML is a GPS tracker[3], which collects the users' location information automatically in the background. Upon starting the application, algorithms analyze the collected data and visualize the results using a map.

Moreover, the app provides its users with statistics on how many kilometers they have travelled as well as at which places, on which journeys, or in which countries they have spent most of their time.

On top of that, the app provides its users with multiple possibilities to provide feedback to the developer: (1) application reviews, (2) a Zendesk[4] client to create

[3] Application that periodically samples the user's GPS coordinate in order to fulfil a task such as collecting data on the user's jogging behaviour.

[4] Zendesk is a SaaS ticket system.

tickets from within the app or the apps' website, (3) a Jira[5] client which allows to comment on and create issues and private emails. At the time of the study, the app had received more than 3000 records of such feedback. Track My Life is available on Android, iOS, and Windows Phone.

3.3 Study Design

As the approach is only eligible for features that can be implemented as some sort of IAPs, the study design requires eliminating non-eligible features first. Features which meet any of the following criteria need to be eliminated from the backlog: (1) bugs, (2) improvements, (3) features with ethical issues, (4) features which do not qualify as IAPs.

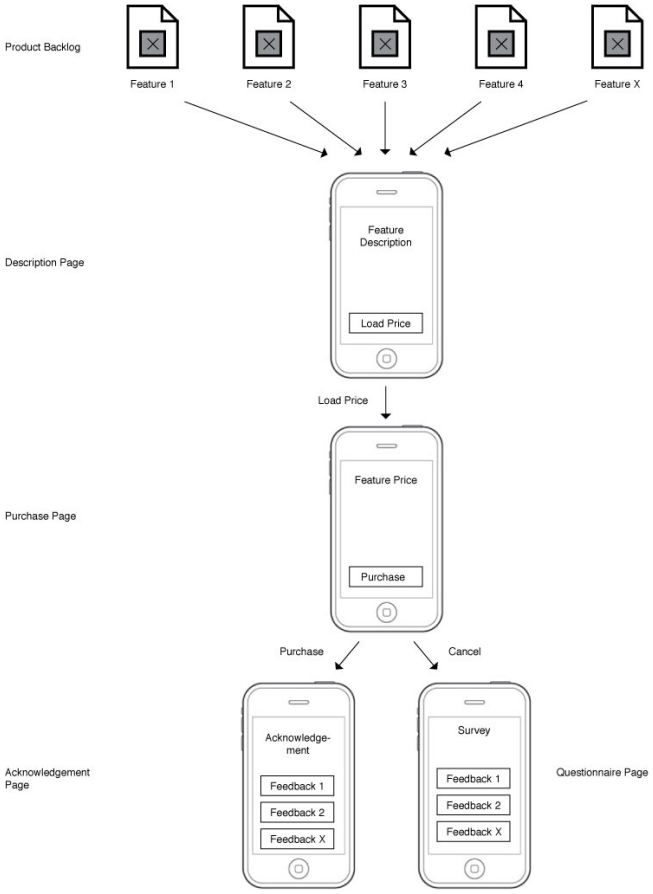

Fig. 1. Approach.

[5] Jira is a defect and issue tracking system.

Figure 1 depicts the different steps of the approach. At first, the product backlog may contain one or more features that are eligible for the approach. Upon starting the app, each user is assigned to only one of the features. Thereby, each user tests only one feature. For each feature, an individual description page has to be designed. The page contains a button that invokes the price-loading routine, as well. From loading the price, the author deduces that the user is interested in the feature. A click on the Load Price button invokes the price to be loaded. In case the software is just available in one currency, the price can directly be displayed. If the software is distributed internationally, the price needs to be converted. After conversion, the price needs to be rounded to realistic values. As an example, after conversion a price might be $1.73 which is an uncommon price. Hence, users might doubt the validity. Therefore, rounding the price to $2.00 is more convenient.

After the price is loaded, users are presented with a purchase button and the price of the feature in their currency which the author calls the purchase page. Depending on whether the user decides to purchase the feature or not, he or she is forwarded to an acknowledgement page or to a questionnaire page.

The acknowledgement page acknowledges to the user that the feature actually has not yet been implemented and that he or she was not charged any money. The research context is disclosed in order not to chase users off. Moreover, it is important to capture the users' feelings towards the methodology in order to exclude disappointed users from future studies.

Therefore, the following scale is used to measure the users' feelings:

- Understand: I understand
- Indifferent: I don't like the approach
- Annoyed: I am annoyed by the approach

Users who choose indifferent or annoyed should be excluded from further studies. Moreover, users can also provide personal messages. Therefore, the following three buttons are suggested, which forward the user to either a feedback form or invoke an email:

- I need the feature urgently
- I want to file a complaint
- I want to say something else

In case the user does not purchase the feature and is forwarded to a survey page (4), the user gets another question in order to find out why users did not buy the feature. Therefore, the following buttons are provided as possible answers:

- It's not that interesting
- It's too expensive
- I don't spend money on apps
- I was disappointed by other purchases
- Write down custom reason

The question on the pricing is especially interesting as it might reveal that there are people willing to purchase the feature at a lower price point.

3.4 Data Collection

The data collection cares about metrics regarding conversion rate and adequate pricing and differences of the features, feedback concerning the research methodology, and the data on why users did not purchase the features.

The following measures were derived for data collection:

Total Number of Purchases per Feature and Price
: The number of purchases that were made per feature and per price. As an example, it is of high interest how many users purchased feature A at price X.

Hypothetical Revenue
: The amount of money that would have been generated in case the feature was a real IAP.

Feelings After Purchase
: How the users reacted after purchasing the feature.

Reasons for Cancellation
: The reasons for which users decided not to purchase the feature.

Organic Feedback
: Whether the approach affects the reviews or other types of feedback that is provided to the app's developer.

3.5 Data Analysis

In order to find the right feature as well as the right price for the feature, the analysis focuses on with which feature at what price revenue can be maximized. Since users who cancelled a purchase are asked whether they cancelled because of the high price of the feature, the author assumes that those users would have purchased the feature at a lower price point.

Moreover, the analysis focuses on how many users are scared off due to the somewhat unethical approach as well. Therefore, the amount of users who understand the approach are opposed to the users who resent the approach.

Fig. 2. Implementation on iOS and Windows Phone.

4 Study Execution

This section focuses on how the author selected the features which were implemented, as well, on the implementation effort related to the approach. Moreover, the course of the study is outlined.

4.1 Feature Selection

As not all features in the backlog are eligible for the approach, the product backlog needs to be filtered. The following characteristics disqualify features: bugs, improvements, features with ethical issues, features which do not qualify as IAPs e.g. internationalisation.

Selection Step	# Features	# Requested
Beginning	40	2127
Bugs Eliminated	25	485
Improvements Eliminated	15	194
Non-IAPs Eliminated	4	41

Table 1. Results of the feature selection.

Table 1 depicts how the features were selected. At the beginning of the study, the product backlog contained 40 elements which were requested 2127 times. The backlog was established by the analysis of different feedback sources such as application reviews, issues which were created by users in the issue tracker, emails by users, or personal messages via the ticket system.

The elimination of bugs from the list decreased the amount of features to 25. After improvements were removed 15 features were left. At the last step, feature requests which are not eligible as IAPs were removed. At the end, four feature requests remained which were requested 41 times.

Out of the four feature requests, two were selected to be implemented. The selection criterion was that the developer considered both features equally important in terms of business value while the implementation effort of one feature was considered drastically higher than the other. Also, both features were requested similarly often.

The two features can be described as follows:

History Feature (HF)
: The history feature enables users to retrace and edit the history of their trajectory.

Statistics Suite Feature (SSF)
: At this point, the amount of statistics implemented in the app is very limited. The statistics suite feature promises to provide users with more statistics.

4.2 Implementation

Figure 2 depicts the implementation of the approach on iOS and Windows Phone. The approach was not implemented in the Android version of the application due to the low amount of users.

The implementation resembles the approach described in 3.3. In order to draw the users' attention to the new feature, a note on the new feature is presented on a very popular page of the app.

Moreover, a mechanism to enable and disable the approach remotely was needed in case the study would result in scaring away many users. Hence, the application makes an http request to a server in order to find out whether the experiment is live or not. Another challenge of the implementation was the currency conversion and adaption of the result to common price application prices in various countries.

4.3 Implementation Effort

The implementation of the approach demanded about one and a half man-days per platform. Implementing the approach itself (i.e. functionality to mock-purchase features as well as the feedback mechanism) consumed most of the time. Adjusting the user analytics in order to derive the required data as specified in section 3.4 was time-consuming as well. Establishing the client-server connection in order to be able to remotely switch the approach on and off did require implementation effort as well. Lastly, packaging the application for release required testing and the usual packaging effort e.g. updating the application description, as well.

4.4 Course of the Study

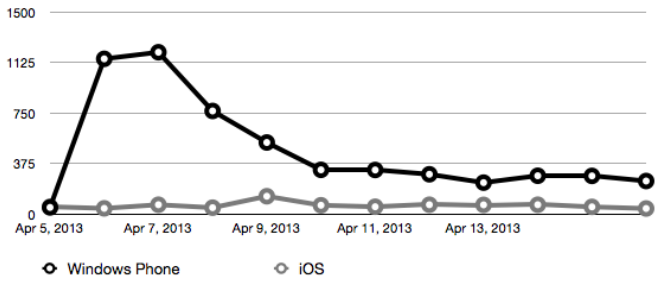

Fig. 3. Course of the Study.

The study started on April 5 on the Windows Phone platform and on April 7 on iOS. Figure 3 depicts the number of daily new users. The number is composed of new users downloading the app on a given day as well as existing users

updating the app to the current version. The study ended on both platforms on April 23. In total, 9426 users participated in the study.

5 Results

The content of this section is twofold: at first the results regarding the feature selection and feature pricing is revealed. Then the data collected regarding the approach is presented.

5.1 Participants

In total, 9426 users have participated in the study. 8501 of those were using the app with Windows Phone, 925 on iOS. 2664 users clicked on the *Check the new Feature Out* button on TML's Statistics Page as shown in figure 2. 1493 showed interest in the feature by loading its price. In total, 294 users did a mock-purchase.

Purchases	SSF1	SSF3	SSF5	HF1	HF3	HF5
# Purchases	**93**	45	33	63	47	22
# Purchases	Statistics Suite: **171**			History Feature: 132		
in Euro	93	136	**166**	63	142	108
in Euro	Statistics Suite: **395**			History Feature: 313		

Table 2. Number of purchases and revenue.

5.2 Purchases

Table 2 depicts the number of purchases in each of the categories as well as the revenue the mock-purchases would have generated. The letters in the columns depict the type of feature, SSF represents the Statistics Suite Feature and HF depicts the History Feature. The number depicts the price point in Euros. Hence, SSF3 depicts the Statistics Suite Feature at EUR 3.

Table 2 depicts that more users purchased the Statistics Suite Feature than the History Feature. Especially at the lowest price point, as well as the highest price point, there are almost 50% more purchases. Therefore, the Statistics Suite can produce more revenue. However, the higher revenue has to be compared to the implementation effort of the feature as well as to other factors such as whether the feature has synergies with other items in the backlog.

Figure 4 depicts the number of purchases and the associated revenue in a column diagram. As anticipated, there is a strong correlation between the price and the number of purchases. The diagram clearly shows that in the case of

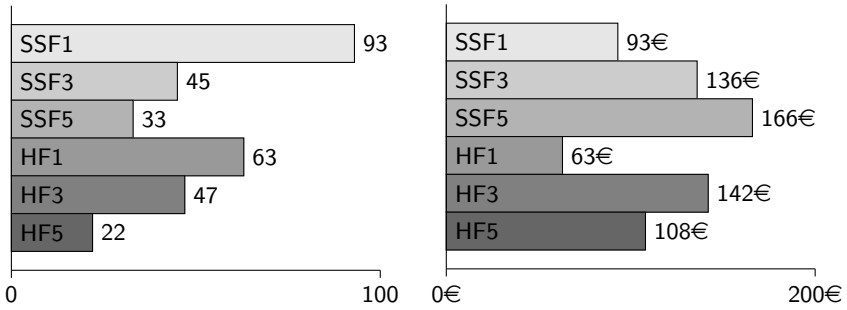

Fig. 4. Number of purchases and revenue.

my features setting the price at either EUR 3 or EUR 5 would result in the highest revenue. The diagram also reveals that users are willing to pay more for the Statistics Suite Feature than for the History Feature. Therefore, the most revenue might be achieved at an even higher price point such as EUR 5 for the Statistics Suite Feature. The adequate price of the History Feature is clearly between EUR 1 and EUR 5.

Feature	SSF1	SSF3	SSF5	HF1	HF3	HF5
Other reason	**38%**	15%	15%	**29%**	20%	14%
Not interested	12%	10%	4%	**16%**	14%	7%
Do not spend money	19%	22%	15%	**23%**	19%	10%
Disappointed	**6%**	1%	2%	4%	2%	1%
Too expensive	25%	51%	64%	28%	45%	**68%**

Table 3. Reasons for which the users cancelled the purchase.

Table 3 depicts the reasons why users cancelled the purchase per category. Most users cancelled the purchase because they considered the feature too expensive. Other than that, many users stated that they do not spend money on apps in general. Very interesting is that only 10% of the users said that they are not interested in the feature. As the number is higher for the History Feature, we can predict that users are more interested in the Statistics Suite Feature. Also very interesting is that 20% cancelled for another reason. However, they were forwarded to a feedback page. Only 0.9% of the users actually submitted feedback. It is also interesting to see, that a small number of users said they did not purchase the feature because they were disappointed with other features they had purchased previously in the app.

Figure 5.2 depicts the feedback that was collected from users who mock-purchased a feature directly after the purchase. The majority of users understood the methodology. 11% disliked the methodology and 3% were very annoyed by the methodology. Only one user selected I need the feature urgently. However, this user was forwarded to a custom message form which he or she did not fill

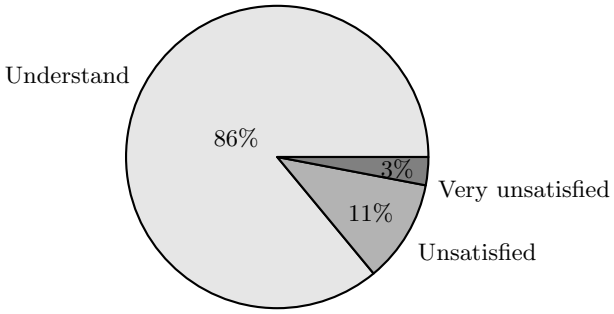

Fig. 5. Feedback after purchasing.

out. No user chose either I want to file a complaint or I want to say something else.
Apparently, some of the very annoyed users were extremely annoyed. Hence, they gave bad reviews. Moreover, the author was approached a few times by users who were exceptionally aggressive. In fact, he was insulted both personally and professionally. For privacy reasons, the letters are not included in this paper.

There were also several reviews by users who obviously did not mock-purchase one of the features but loaded the price. As a result, they wrote bad reviews because of the too high pricing. As an example, one user commented:

> So close to being worthwhile, yet just sad not fulfilling in its potential, and 6 bucks for the good stuff?? Insanity

As user comments are essentially important for smartphone applications, choosing a price which does not achieve the maximum revenue but balances the trade-off between happy users and feature price in a more convenient way might be a good choice. In order to minimize the number of users who dislike the approach or are very annoyed by it, there should be some benefits for users who participated in the approach. As an example, a price reduction once the feature is finally released. On top of that, users who dislike the approach or are annoyed should be excluded from further studies.

5.3 Interpretation

In order to balance between highest revenue and customer satisfaction, the author of the app considers the EUR 3 price point to be the most adequate price for both the Statistics Suite Feature and the History Feature.
Moreover, the author would implement the Statistics Suite Feature as it obviously results in higher revenue and meets the app's general roadmap better than the History Feature although the implementation effort is higher.

The results also match the demand for SSF that was identified during the feature selection process as SSF was requested more frequently than HF.

Specifically with smartphone applications, users are not accustomed to purchasing IAPs for EUR 5. However, in different contexts such as desktop appli-

cations, higher prices are usual. Therefore, users might not leave bad comments due to high pricing in desktop applications.

6 Discussion

The results have shown that the approach allows testing the business value of different features as well as how often they are purchased at various price points. However, the test comes at the price of potential customer dissatisfaction and other interaction effects (such as implications on customer ratings that might influence future buying decisions). The essential limitations are the following:

6.1 Applicability

The approach is only applicable for features that can be implemented as IAPs. In the presented case, most requirements that were extracted from user feedback were either bugs, improvements, or other non-IAP-eligible requirements.

6.2 Validity for Larger Development Teams

The context of the study is an app that is developed by a single developer. Hence, prioritization issues that are common in larger development teams such as disagreements between team members did not occur in the execution of this study. Therefore, the results may vary in larger development teams.

However, we assume that the core of the presented prioritization approach applies in larger teams as well.

6.3 Validity of the Identified Ideal Price Points

In the context of this study, a price is considered *ideal* in case it yields the highest revenue at the fewest concerns regarding customer satisfaction, app reviews, and user retention. However, these criteria are very smartphone app-specific. Hence, the criteria for an *ideal* price point may vary in different contexts.

Moreover, the authors of the paper concluded the *ideal* price points based on the collected data. Hence, the price points were identified using empirical data rather than theoretical grounding.

6.4 Legal & Ethical Issues

Offering a feature that had actually not been implemented can be compared to selling something in a store catalogue which is actually not in stock. This might be seen as not correct ethically and there might also be legal issues attached to it. On top of that, the approach might interfere with smartphone operating

system manufacturers' guidelines although neither Apple's[6], nor Google's[7], nor Microsoft's[8] guidelines cover the topic.

7 Conclusion & Future Work

This paper introduces an approach that allows finding out about the real revenue of features which had actually not yet been implemented. Moreover, it provides insights and decision support on setting an adequate price for a feature. The approach was implemented in a smartphone application on two operating systems. Almost ten thousand users participated in the study.

The results are promising as they show that the approach indeed allow us to gain insights about feature revenue and pricing. However, it comes at the cost of customer dissatisfaction so that we recommend using it only for important feature implementation decisions (e.g., selecting features with significant differences in implementation cost). This serves as a motivation for future work on the approach. One way to address the customer dissatisfaction could be to provide mock-purchasers with a benefit e.g. a price reduction. On top of that, dissatisfied users could be treated in a special way, e.g., by excluding them from further studies.

The execution of the study unveiled that there is a non-negligible effort to conduct the study. First of all, the approach required sound judgments about what data has to be collected throughout the study. Second, the implementation was time-intense as the data collection needed to be implemented but also the smartphone operating systems' organic IAP processes needed to be emulated. As an example, the feature price needed to be converted and adapted to the operating systems' usual pricing thresholds. On top of that, a mechanism to both remotely start and stop the study was needed. Lastly, a mean of collecting the feedback and providing feedback forms was required. As a result, creating a boilerplate for other software developers in order to efficiently make use of the approach serves as motivation for future work.

References

1. Bakalova, Z., Daneva, M., Herrmann, A., Wieringa, R.: Agile requirements prioritization: What happens in practice and what is described in literature. In: Requirements Engineering: Foundation for Software Quality. Volume 6606 of Lecture Notes in Computer Science., Berlin, Germany, Springer Verlag (March 2011) 181–195

[6] `https://developer.apple.com/appstore/resources/approval/guidelines.html#purchasing-currencies`

[7] `https://play.google.com/about/developer-content-policy.html`

[8] `http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh184843(v=vs.105).aspx`

2. Bakalova, Z., Daneva, M., Herrmann, A., Wieringa, R.: Agile requirements prioritization: What happens in practice and what is described in literature. In Berry, D., Franch, X., eds.: Requirements Engineering: Foundation for Software Quality. Volume 6606 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2011) 181–195
3. Halkjelsvik, T., Jørgensen, M.: From origami to software development: A review of studies on judgment-based predictions of performance time. Psychological Bulletin **138**(2) (2012) 238–271
4. Lacey, M.: Prioritization. Website (January 2012) Available online at `http://msdn.microsoft.com/en-us/library/hh765981.aspx`; visited on April 17th 2013.
5. Buhl, H.U., Kundisch, D., Schackmann, N., Renz, A.: Spezifizierung des kano-modells zur messung von kundenzufriedenheit (2006) Available online at `http://www.wi-if.de/paperliste/paper/wi-142.pdf`; visited on May 7th 2013.
6. Galvis Carreño, L.V., Winbladh, K.: Analysis of user comments: an approach for software requirements evolution. In: Proceedings of the 2013 International Conference on Software Engineering. ICSE '13, Piscataway, NJ, USA, IEEE Press (2013) 582–591
7. Wiegers, K.E.: First things first: Prioritizing requirements. Website (1999) Available online at `http://www.processimpact.com/articles/prioritizing.html`; visited on September 9 2013.
8. Runeson, P., Höst, M., Rainer, A., Regnell, B.: Case Study Research in Software Engineering: Guidelines and Examples. John Wiley & Sons (2012)