Rapid Lean UX Development Through User Feedback Revelation

Frank Elberzhager[1(✉)], Konstantin Holl[1], Britta Karn[2], and Thomas Immich[2]

[1] Fraunhofer IESE, Fraunhofer Platz 1, 67663 Kaiserslautern, Germany
{frank.elberzhager,konstantin.holl}@iese.fraunhofer.de
[2] Centigrade GmbH, Science Park 2, 66123 Saarbrücken, Germany
{britta.karn,thomas.immich}@centigrade.de

Abstract. The development of software within short timeframes calls for concepts like minimum viable products with lean development. An agile development setting allows software products to be put on the market in time. Nevertheless, quality, especially in terms of user requirements, suffers when the focus is on the speed of the development. Therefore, we have developed the approach Opti4Apps, which considers user feedback automatically. This automation enables rapid user feedback to be revealed, which is needed for lean development in order to achieve high software quality in accordance with the users' needs. This paper shows how the approach can be applied smoothly in agile development settings by analyzing common agile practices with regard to our user-centric feedback approach Opti4Apps. It turned out that with most practices, the additional effort is low, and the positive influence can be highly beneficial.

Keywords: Quality assurance · User experience · Lean development · User-centered · Feedback · Agile practices

1 Introduction

With the rise of applications that have a short time to market, quality has often become subordinated to features. Since it is both relevant for companies to be the first on the market and to increase the quality of mobile applications, ensuring high quality is moving into the focus of development to help companies remain competitive. To achieve this quality, companies need to invest in quality assurance strategies and define new priorities [1]. This requires the investigation of product and process quality in the context of agile and rapid software development.

To improve current practices for quality assurance in agile settings, we focus on early feedback from users in this publication. According to our observations from practical environments, but also in line with other researchers, users are often not given the priority they deserve, and companies struggle when it comes to gathering and analyzing user feedback efficiently [6]. However, on the other hand, feedbacks are a rich source for improving the software products. In order to make this efficient, automation should play a role. A reasonable instrument for realizing such a procedure is a semi-automated feedback elicitation, analysis, and processing framework, so the effectiveness and

M. Felderer et al. (Eds.): PROFES 2017, LNCS 10611, pp. 535–542, 2017.
https://doi.org/10.1007/978-3-319-69926-4_43

efficiency of early user feedback consideration during further development can be examined with the goal of assuring the quality and acceptance of a mobile application developed in a minimalistic way [2].

While MVP development enables early feedback and very fast time to market, the quality of the resulting product suffers in comparison to traditional development. In contrast, with traditional development, there is no early feedback and time to market is longer.

Our approach Opti4Apps for automated consideration of user feedback could realize and extend the benefits of MVP development through the development and use of a framework based on the automatable elicitation and analysis of feedback as well as through the use of an effective and efficient quality assurance methodology. This rapidly focuses the lean development on the user's requirements. Furthermore, the feedback and the insights from one development could be reused in parallel or subsequent developments.

Nevertheless, in order to reap the benefits of automated consideration of user feedback, compatibility with existing agile processes is required. Therefore, we show the compatibility of a selected set of common agile practices with our Opti4Apps approach.

This article is structured as follows: Sect. 2 describes the related work, in particular basic concepts, prior work, and an agile references process. Section 3 presents top-down the feedback-based lean UX development process followed by the assessment of selected agile practices. Section 4 wraps up the results and experiences followed by ideas for future work.

2 Related Work

2.1 Lean UX

The development of mobile apps calls for concepts like minimum viable products produced in lean development due to short timeframes. The Lean UX approach is a concept that combines three development methodologies: design thinking, agile development, and Lean Start-up [5].

1. The principles of design thinking show that design methods can be used in every phase of a project from any discipline. Non-designers should be encouraged to use design methods based on this approach, as it supports teams in collaborative design across roles. Furthermore, it is a customer-centered approach as it takes not only the user needs into account but also the technological possibilities and the business view.
2. The second methodology in Lean UX is agile development. It has been an important approach for software developers for a long time, especially the Scrum process. The important elements of Lean UX are the iterative and incremental approach, which enables the team to respond to change immediately (compared to the waterfall model), collaboration in teams and with the customer to ensure continuous feedback, and a strong communication culture.
3. The third methodology is Lean Start-up. The basis of this approach is the "build-measure-learn" loop, which helps teams to minimize project risks and supports quick

feedback and faster completion of projects. This faster completion results from the Minimum Viable Products (MVPs) that the teams are building to get user feedback as soon as possible with the help of rapid prototyping. According to Lean UX [5], an MVP is, on the one hand, the smallest thing that helps to test assumptions with the help of a prototype or other product developed by the team. In this case, the MVP is built to learn something and the team benefits from it, but there is no immediate benefit for the user. On the other hand, an MVP in Lean UX is defined as the smallest version of the end product that is delivered to the users and addresses a problem or need the user has. In this case, the focus of the MVP is on the benefit for the user. In the Opti4Apps context, we define MVP in the latter way: a product that is usable and valuable for the user who benefits from it.

The Lean UX approach of continuous learning and testing leads to the following principle, which also defines the start of a Lean UX project: Assumptions before requirements! The first step in the Lean UX process is to declare assumptions that should be validated along the process.

2.2 Prior Work

Figure 1 provides a conceptual overview of the previously defined user feedback approach [2]. There is a mobile application as an MVP on a mobile device and users who use the application. The users can give feedback. This feedback comprises the application's usage data (e.g., usage frequency, duration, or misentries), state (e.g., installation and online state), and explicit user feedback (reviews, bug reports). Such feedback can be provided by the user automatically (e.g., via a specific agent running on the mobile device), semi-automatically (e.g., some data is tracked by the mobile device and has to be sent manually to a backend), or manually (e.g., users provide some written feedback in the app store). In other words, feedback can be provided explicitly by the user or implicitly through measurement by technical means.

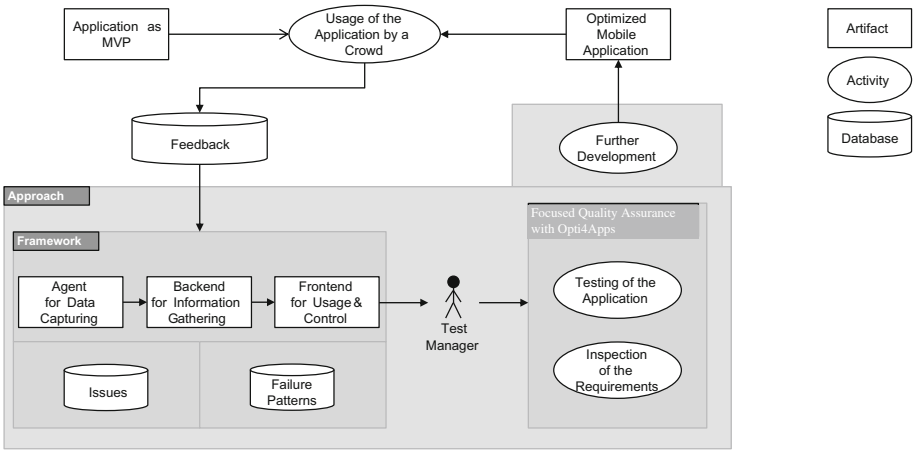

Fig. 1. Automated user feedback consideration in the Opti4Apps approach.

The feedback then has to be edited, analyzed, and provided in a suitable way so that the company developing the application gets information it can use to improve the application. A framework is used for this purpose. It uses, for example, the data provided by the agent integrated into the mobile application, direct feedback, or feedback gathered by performing data mining analyses to detect existing deficiencies or reveal improvement ideas. All such data is consolidated by a backend and classified to generate a more suitable overview.

The analysis of the collected information in the backend via data mining is intended to enable a fast learning effect with respect to existing deficiencies. Thus, it can provide a baseline for effective further development as well as for focused quality assurance. Identified failure patterns (based on mobile-specific failure classes [3]) can increase the effectiveness of the quality assurance of the current development. Because of the reusability of failure patterns, these can also be used for parallel and subsequent developments.

One of the main aspects of the framework could be the recognition of usage patterns that may be considered as failures as part of the mentioned failure patterns, which constitute a collection of typical failure causes and impacts in the area of mobile applications.

The control of the framework and the utilization of the produced information result in several role-specific tools, which form the frontend. A dashboard with different views depending on the data analysis is used by a test manager. This role is responsible for taking further actions based on the results of the feedback analysis. The main tasks include controlling quality assurance (i.e., deriving new test cases based on the identified problems) or sharing the results with a requirements engineer in case new feature wishes are identified.

By using the failure patterns during the inspection of the requirements specification as well as during testing of the mobile application, focused quality assurance is established, thus optimizing the mobile application. Based on the user feedback, the optimized mobile application is again distributed to the users. This is realized by a distribution software installed on the device (e.g., HockeyApp, TestFlight).

2.3 Agile Reference Process

Mobile applications are typically developed in short, often agile, development cycles [4]. The agile development process is often performed according to the project management framework Scrum. The overall objective of Scrum is to enable the development team to react quickly, simply, and appropriately as part of the development process. Short time to market can be achieved better than if classical development approaches are used. Time-consuming activities like updating outdated plans should be avoided.

As part of Scrum, all known requirements are stored in a product backlog. This provides the input for the sprint backlogs. Development with Scrum occurs in iterations. These iterations are called sprints. The output of each sprint is intended to be a working product increment, respectively a potentially shippable product.

Scrum does not prescribe which techniques have to be used during development. This is mostly up to the development team [7]. Neither does Scrum dictate the types of

tests that have to be performed [4]. Independent of the specific technique selected as part of Scrum (e.g., Extreme Programming, Test First), testing in agile development can be assigned to the fundamental test process, just like testing in classical software engineering.

3 Process Integration

3.1 Conceptual Picture of Feedback-Based Lean UX Development

The Lean UX framework is our starting point. The UX team starts an MVP project with a pre-process. Elements of the pre-process are, e.g., Scoping, User Research, Conceptual Design, and Design Engineering Workshops. The exact content of this pre-process is not determined by this model; rather, the objective is to derive user stories from user needs, which will be the working basis in the following sprint process. Hence, it is necessary to clarify user needs that are evaluated based on real user feedback. Those user stories that are most relevant for the current scope are taken into the following sprints and form the basis for the resulting MVP. It is possible to run parallel processes for multiple MVPs with different teams as well as to operate sequentially. In addition, it is possible to develop the same user story in different ways on different tracks, e.g., to do A/B-testing later on. Of course, it is important to continuously consider the "big picture", i.e., the project as a whole, to ensure that it will be possible to consolidate the results of different MVPs in one overall product or system.

The objective of the lean process for Opti4Apps is to gain user feedback at any time of the process, which is shown in the area below the model, despite the short timeframe. The classical feedback approaches make heavy use of concept testing (which serves the designer) and mostly will not work in this timeframe, as there are too many iterations. With Opti4Apps, the user feedback is gathered during product testing by a tracking agent that automatically obtains relevant usage data from users.

Nevertheless, this model is initially an idealization and does not consider all "real-world circumstances" of projects. The process must be adapted depending on diverse influencing factors, for example team size and constellation, project domain, "type" of customer (internal, external), and the take-off point of the project (an all-new development vs. a refinement of an existing product). Many other influencing factors are conceivable. All these factors determine what a more detailed view of the model would look like. In order to understand how different instances of agile processes behave when such a user-centric approach is considered, we analyzed common agile practices with respect to invasiveness and benefit. The results are presented in the next section.

3.2 Assessment of Selected Agile Practices

In order to make the lean UX process more concrete, we analyzed how common agile practices cooperate with Opti4Apps and what the potential benefit is when Opti4Apps is applied in an agile process (i.e., if agile practices are followed). Two experienced researchers (more than 10 years of experience) performed an evaluation of the most common agile practices. We addressed 11 top-level topics from Diebold and Dahlem

[7] (some of the 18 originally mentioned ones have no connection to user feedback, such as refactoring), and considered a total of 21 concrete agile practices. We evaluated invasiveness and benefit on a four-point scale (see legend in Table 1):

- Invasiveness: When Opti4Apps is applied, how much is the agile practice influenced (under the assumption that the agile practice is applied) and how much adaptation may be necessary (which might result in higher effort to apply the agile practice)?
- Benefit: How strong does the agile practice (and indirectly the overall agile development process) benefit when Opti4Apps is applied?

Table 1. Invasiveness and benefit rating of using agile practices with Opti4Apps

Top-Level Topic	Agile Practice	Invasiveness	Benefit	Remarks
Quality Check	Pair Programming	⇘	⇘	Feedback from pair
	Code Review	⇘	⇘	Suggestions from Opti4Apps framework
	Usability Expert Review	⇘	⇨	Defect patterns from Opti4Apps framework for review
Customer Involvement	Product Owner	⇧	⇧	User researcher role, controlling Opti4Apps framework
Validation	Test-driven Development	⇨	⇨	Defect patterns from Opti4Apps framework for test
	Explorative Testing	⇘	⇧	Suggestions from Opti4Apps framework
	Crowd Testing	⇧	⇧	Systematic crowd test and automatic clustering/interpretation
Learning Loop	Retrospective	⇘	⇨	Dashboard shows quality details / feature requests
Planning Meeting	Planning Poker	⇘	⇩	Maybe more user stories lead to more complex planning poker
	Feature Freeze	⇩	⇩	-
	Jour Fixe	⇘	⇨	Dashboard shows quality details / feature requests
Progress Monitoring	Burn Chart	⇩	⇩	-
	Definition of Done	⇩	⇩	-
Product Vision	Product Backlog	⇘	⇧	More features, more bug reports
Specification	Sprint Backlog	⇘	⇧	More features, more bug reports
	Personas	⇘	⇧	New information about stakeholders / customers
	Design Thinking	⇨	⇨	Automated feedback regarding early prototypes
	User Stories	⇘	⇨	More user stories with requested features / bug corrections
Continuous Delivery	Continuous Integration	⇩	⇘	Supports prioritization
Frequent Releases	Code Generation	⇩	⇩	-
Daily Discussions	Standup Meeting	⇘	⇨	Dashboard shows quality details / feature requests

Legend: high ⇧, medium ⇨, low ⇘, none ⇩

Let's consider the topic "Quality check" as a first example to understand the rating: During pair programming, a second person watches what is being programmed, and can give direct feedback. When the feedback cannot be implemented directly, it can be documented in the Opti4Apps framework. This requires little effort (=low invasiveness), but the benefit from such feedback is also rather limited as the amount of such additional feedback is expected to be low. For a code review, experiences regarding typical issues stored in the Opti4Apps framework can be used and checked. However, their number is again expected to be low. During a usability review, defect patterns might be used to control the review, which makes it more effective. The level of invasiveness is again low, as just some information is consumed, but the benefit is, on average, medium.

Besides crowd testing, customer involvement is the only agile practice that has a high level of invasiveness. The reason is that either a new role is needed (a user researcher who controls the Opti4Apps tasks) or that a role such as the product owner has to perform it, which also results in some effort. However, the benefit of such a dedicated role, respectively the product owner who owns the relevant tasks, is highly

beneficial, and to a certain extent forms the core of Opti4Apps, as this role controls, for example, how to handle all the feedback and determines consequences resulting from the analyzed feedback.

There are also agile practices that have no influence (and usually no benefit), for instance burn charts. It is simply a mechanism for visualizing the current status, but new feedback has no direct influence on this practice.

Certain practices support the same benefits. For example, a retrospective, a jour fixe, and a standup meeting are all influenced in a minimal way by Opti4Apps (mainly due to some more feedback which needs little extra time to mention), but have at least a medium positive influence due to new feature ideas or bug indications that may be revealed. There also exist further agile practices that we did not consider in our analysis; however, we picked a set of the most common practices [7] to start our analysis. Table 1 shows the complete evaluation results together with a short explanation of every agile practice.

Of course, the individual ratings can be further discussed and might lead to adaptations depending on the concrete context. On purpose, we did not use a number schema, but tendency arrows, which indicate the general evaluation direction for every agile practice. The agile practices should also be assessed by other researchers and practitioners in order to get a more stable evaluation. Moreover, Opti4Apps should be applied in concrete agile development settings in order to get higher confidence in the initial rating. The current evaluation is mainly based on arguments and on our own experience from several development environments, and serves as a starting point for initial discussions and further evaluations. However, though slight adaptations are reasonable, the trends will probably mainly remain.

The evaluation results show that Opti4Apps is compatible with many agile practices, and that the additional effort or need for changes (expressed as invasiveness here) is rather low. Of course, any new methodology requires some investment effort, but considering the benefit that customer feedback provides in terms of new features and features that are really expected by customers, as well as in terms of indications about quality issues, it is worthwhile the effort.

Practitioners might use the evaluation of these agile practices to check how much Opti4Apps influences their concrete agile development process, and to get further ideas regarding which practices to use in order to gain an even higher benefit and get concrete feedback from customers for further development. Researchers can further analyze agile practices and check whether our rating fits in other settings.

4 Conclusion and Future Work

In this article, we again took up the challenge of how software-developing companies can consider user feedback more strongly. To this end, we provided an overview of our Opti4Apps approach, which gathers user feedback from several sources automatically. Modern software development tends to become ever faster, with trends such as continuous delivery and DevOps. At the same time, it still has to ensure high quality and develop those features that users really demand. To deal with this situation, we

introduced our Opti4Apps approach and showed its compatibility with several agile practices. The approach does, of course, require some investment; however, it turned out that the challenge of really considering the user during development can be highly supported by our approach without large investments or changes in the agile processes.

In the future, we will substantiate our initial classification and rating of agile practices by discussing them with more experts, but also by observing real applications of the Opti4Apps framework in agile developments. We are convinced that this will contribute to stronger consideration of the user during development, as intended by the Agile Manifesto, and that our approach will provide specific guidance for practitioners.

Acknowledgments. The research described in this paper was performed in the project *Opti4Apps* (grant no. 02K14A182) of the German Federal Ministry of Education and Research (BMBF). We also thank Sonnhild Namingha for proofreading.

References

1. Buenen, M., Teje, M., Carrel, I.: Testing and SMAC Technologies: Ensuring a Seamless and Secure Customer Experience. World Quality report 2014–2015, Sixth edn., Capgemini, HP, Sogeti (2014)
2. Holl, K., Elberzhager, F., Tamanini, C.: Optimization of mobile applications through a feedback-based quality assurance approach. In: 15th International Conference on Mobile and Ubiquitous Multimedia, Finland, pp. 1–3 (2016)
3. Holl, K., Elberzhager, F.: A Mobile-specific failure classification and its usage to focus quality assurance. In: 40th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2014), Italy, pp. 385–388 (2014)
4. Linz, T.: Testing in Scrum: A Guide for Software Quality Assurance in the Agile World, 1st edn. dpunkt.verlag, Heidelberg (2014)
5. Gothelf, J., Seiden, J.: Lean UX: Applying Lean Principles to Improve User Experience. O'Reilly, Sebastopol (2013)
6. Fabijan, A., Olsson, H.H., Bosch, J.: Customer feedback and data collection techniques in software R&D: a literature review. In: Fernandes, J.M., Machado, R.J., Wnuk, K. (eds.) Software Business ICSOB 2015. LNBIP, vol. 210, pp. 139–153. Springer, Cham (2015). doi: 10.1007/978-3-319-19593-3_12
7. Diebold, P., Dahlem, M.: Agile practices in practice: a mapping study. In: 18th International Conference on Evaluation and Assessment in Software Engineering (2014)