GESELLSCHAFT
FÜR INFORMATIK

Fachausschuss
Management der Anwendungsentwicklung und -wartung (WI-MAW)
im FB Wirtschaftsinformatik

Jahrgang 27 Heft 1
ISSN 1610-5753

März 2021

Schwerpunktthema: Datengetriebene Anwendungen und Innovationstreiber im Projektmanagement zukunftsfähiger Organisationen
Martin Engstler, Masud Fazal-Baqaie (Hrsg.)

Inhalt

45. WI-MAW-Rundbrief

Entrepreneurial Software Engineering: Towards a Hybrid Development Method for Early-Stage Startups

Daniel Brunner[1], Jürgen Münch[2], Marco Kuhrmann[1]

[1]University of Passau, Innstr. 33, 94032 Passau, Germany;

danielbrunner2007@web.de, kuhrmann@acm.org

[2]Reutlingen University, Danziger Str. 6, 71034 Böblingen, Germany,

j.muench@computer.org

Abstract: A considerable share of innovative software-intensive products is developed by startups. However, product development in an early-stage startup is not a sequential process. A business idea is usually based on a number of assumptions. The riskiest assumptions need to be tested. Depending on the test results, a product strategy may change several times. This raises the question of how to create sufficiently stable software using engineering principles despite a dynamic product strategy that is subject to many uncertainties. Hybrid development methods that combine agile aspects with classical engineering methods seem to be a good choice in such a start-up context. This paper proposes a lightweight hybrid development method that provides early-stage startups with a framework to support the development of single-feature minimum viable products. The method was derived from a start-up company's founding case and evaluated in expert interviews. The proposed method is intended to provide a basis for discussion between practitioners and scientists with the aim of better understanding the application of software engineering principles in software start-ups.

Keywords: Software Startup, Hybrid Method, Guideline, Method Proposal, Lean Startup, Entrepreneurial Software Engineering

1 Introduction

In the last decade, a number of disruptive software startups emerged [Gu15], and many of these startups grew and revolutionized the market, e.g., Flixbus, Uber, and AirBnB. Furthermore, in several fields, former startups have become market leaders, e.g., Amazon, Google, and Facebook. Startups contribute significantly to the world economy, and, through the accelerating digitalization, their impact will further grow.

An important area for startups is the mobile software market. Due to the increased use of smartphones and tablets in web-based environments, many new product ideas are developed and realized as *Apps*. Since there are rich and mature development frameworks available, the development of apps is a straightforward and fast approach to make an innovative idea reality, which improves the level of "attractiveness" of startups. Creativity is in the spotlight; there are no large hierarchically organized teams, and the main thing that matters is the product. However, even though there are easy-to-use tools available, "easy" is a relative term. Still, software development is a challenging business and developing high-quality software requires the skilled professionals.

Startup Challenges. Wassermann [Wa16] stated that software development in startups is often more "hacking" than a systematically implemented practice. Together with the high degrees of technological uncertainty and a way to fast – and often not reflected – implementation of "agile" methods, many startups fail. For instance, Giardono et al. [Gi+14] report that more than 60% of the startups fail within their first five years. Software development, notably sustainable software development, requires some organization. Once the product is in the market, increasingly complex processes need to be installed, e.g., for feature development, change management, bug fixing, and general innovation activities – not to forget the organization framework including, such as acquisition or sales.

Besides, a major challenge is that the business idea is based on many risky assumptions, e.g., is the problem to be solved an actual problem in the target customer segment and, moreover, is it worth solving? If one of the assumptions does not hold, the startup's business model is at stake. Therefore, risky assumptions must be tested as early as possible to adjust the product strategy accordingly. Quite often, a mockup or a software prototype, a so-called *Minimal Viable Product* (MVP), is used to test assumptions in the problem domain. At this point, software engineering principles are often neglected, since understanding the problem domain is more important than developing a sustainable software product. Yet, the software engineering principles become important once the product has been launched [Wa16].

This is where *Entrepreneurial Software Engineering* enters the stage. Entrepreneurial Software Engineering is concerned with the following question: *How to efficiently and effectively develop software-intensive systems as part of an entrepreneurial or innovation process?* A key concern in this regard is to ensure from the very beginning on that a process is used that allows for developing a product that is robust from the software engineering point of view. That is, for example, just in the development of the initial product (parts), the questions for the product's architecture must be asked, and if it is possible at all to design a solid architecture without knowing exactly what the final product will look like and which features it will have? Eventually, it is a matter of trade-off decisions, since there are different goals in the different phases of a startup or innovation process, ranging from testing ideas early over testing solution alternatives to ensuring high quality of the final product in the market. Entrepreneurial Software Engineering aims at appropriately supporting the different phases, at creating synergies between the development activities in these phases, and at minimizing waste.

Contribution and Outline. This paper proposes a lightweight hybrid method tailored for use in early-stage startups that are focusing on *single-feature MVPs*[1]. The method was inspired by the needs becoming obvious in the *Zippr* startup, by the principles of lean software development, and the concepts described in experiment-driven software development and continuous experimentation. The method was created using data about combined method and practice use, and the method was evaluated in expert interviews. The method provides an initial framework to be modified and extended by practitioners in their various contexts.

The remainder of the paper is organized as follows: Section 2 provides an overview of the background and related work. Section 3 introduces the proposed method in detail and provides details on the method's evaluation. The paper is concluded in Section 4.

2 Background and Related Work

There are several definitions of the term "startup" [BD12, Ri11, Su00]. For instance, Blank et al. [BD12] define a startup as a "temporary organization designed to search for a repeatable and scalable business model", and Ries [Ri11] defines a startup as "a human institution designed to deliver a new product or service under conditions of extreme uncertainty". Uncertainty arises from a variety of causes [Gi+14]: startups often do not know their customers or the markets. Often, startups offer solutions to problems that many customers were not even aware of [LM16]. Another issue is the rapid evolution of startups. Once the business model is identified, scaling the startup fast becomes the main focus. Other than established companies, startups can (usually) adopt quickly to external influences. However, a lack of resources is often found that hinders grow, e.g., limited human and physical resources [Gi+14]. Also, young venture team members are usually inexperienced and have too little (software) engineering skills. Startups usually aim to deliver their products to customers as fast as possible to establish themselves in

[1] This paper is based on the Bachelor Thesis "A Hybrid Development Method for Early-Stage Start-Ups" authored by Daniel Brunner at the University of Passau. The paper provides the proposed development method in an improved straightforward way. Further details, especially the initial evaluation of the proposed method, due to page limitations, has to be taken from the Bachelor Thesis.

the rapidly changing and progressing markets, which are under heavy competition. Reducing speed of the product delivery might lead to competitors releasing comparable solutions. To speed up the product delivery, many startups focus on one single product, whereas established companies offer a variety of different products in several horizontal and vertical markets.

Lean Startup. The *Lean Startup* concept was introduced by Ries [Ri11] in 2011. Lean Startup is based on customer-focused and agile software development, and implements Lean management techniques to guide the creation of new businesses and products. A key component is the set of *lean principles*, e.g., avoid waste to reduce the effort required for the product development. The principles can be implemented by combining business-driven hypothesis, experimentation, and iterative product releases. Especially iterative development requires deep knowledge of the needs of early customers – also often referred to as early adopters. Putting the customer into the spotlight can lead to reduced market risks, e.g., to avoid launching products that do not provide value.

Minimal Viable Products. The most relevant concept of the Lean Startup Principles in the context of this paper is the *Build-Measure-Learn* cycle (BML). The BML-cycle aims at turning assumptions about the product or its customers into knowledge that helps deliver valuable solutions. The BML-cycle helps increase the speed of the product development through fast feedback loops. In the *Build*-phase, testable assumptions – hypotheses – are posed that usually reflect the startup's vision of the product and business strategy. A *Minimum Viable Product* (MVP) is built, which is a "version of a new product, which allows a team to collect the maximum amount of validated learning(s) about customers with the least effort" [Ri11]. There are various types of MVPs, which can be used depending on the case or vision [DA16], e.g., landing pages, mockups, and single-feature MVPs.

Related Work. In the last decade, research interest in the field of startups increased. However, literature on software development activities in startups is scarce. Few literature reviews have been conducted on different startup-related topics. For instance, Paternoster et al. [Pa+14] conducted a systematic mapping study on software development practices. They found that startups tend to select and use development practices opportunistically. Klotins et al. [KUG15] identified development practices startups use and mapped findings to the *Software Engineering Body of Knowledge* (SWEBOK) knowledge areas. They found 11 out of 15 knowledge areas are covered by research. Zettel et al. [Ze+01] were among the first that developed a lightweight software development process for startup companies (the so-called LIPE process). They focused on the development of e-business applications and their process is widely based on Extreme Programming. The aim of this process is to exploit the long-term benefits from the application of software engineering methods and principles, and to use some of them from the very start. Besides, research did not provide guidelines for software development for any stage. Yet, Berg et al. [Be+18] found a shift in the research focus of software startup research. While most research is conducted within the SWBOK knowledge areas, increasing interest could be found in the process and management areas. Recently, Tegegne et al. [TSA19] studied the methods and practices startups use for their software development, and Giardino et al. [Gi+14] examined practices that are commonly applied in startups. Both studies showed that startups tend to use agile and Lean Startup methodologies to keep flexibility and to be able to adopt fast to the market and customer needs. They tend to use lightweight, often informal and customized methods [TSA19], which are often focused on iterative and incremental coding – Wasserman [Wa16] describes these methods as "low-ceremony processes". However, strictly following a method is uncommon in software startups due to limited resources and time pressure. Startups often select practices considered relevant from known methods and adapt these to the individual needs, which is referred to as a *hybrid method* [Ku+18]. Nevertheless, startups must also be prepared for the future, since without adequate processes, failure in the long term is likely [Cr02, GWA14, Wa16]. Startups have to balance the fast validation of the business model and provision of a high-quality product, which is often referred to as "Developers Dilemma"

[TSS16]. Having a solid framework in place, which can be evolved over time, reduces the risk of inadequate processes.

In Lean Startups, experimentation is used to validate product and business assumptions to seek a valuable product, and continuous value delivery is key. *Continuous experimentation* helps identify the features to build and deploy. Fagerholm et al. [Fa+14, Fa+17] introduced building blocks continuous experimentation and the "Stairway to Heaven" model describes an evolution path companies can take to be equipped for continuous experimentation [OAB12]. For conducting continuous experimentation within a company, Fagerholm et al. [Fa+17] introduced the *RIGHT* model, and Olsson and Bosch [OB14] propose the *HYPEX* model. Both models aim to reduce opinion-based decision-making and foster informed decision-making grounded in empirical evidence, e.g., obtained in (feature) experiments.

3 A Hybrid Development Method for Early-Stage Startups

This section presents the hybrid development method for early-stage startups. It starts with presenting the core requirements of the model, before the model as such is presented.

3.1 Requirements

As outlined before, software startups develop software under special conditions, notably in interdisciplinary teams in which the software development expertise is not necessarily strong. Therefore, such team require methodological and technical support. Form this assumption, we derive the following core requirements:

R1. The development method must be lightweight.
R2. The development method must allow for later scaling (once the product matures).
R3. The development method must include the continuous experimentation paradigm.
R4. The development method must support the fast and cost-effective MVP development.
R5. The development method must ensure that technical debt is avoided.

These core requirements are straightforward: the development method must be lightweight to provide an easy start for creative teams, but it must also allow for scaling. Once the product is in the market, further processes need to be installed, e.g., evolution processes and further business processes that need to be aligned with the software development. In order to define new requirements and features, and to identify those that potentially generate the most value, continuous experimentation should be included. Among other things, continuous experimentation should also be integrated with the cost-effective development of MVPs. Finally, related to R2, the development method should ensure that the software developed fulfills basic quality requirements. The goal is to avoid as many "quick hacks" as possible that need to be costly fixed later.

The development method, which is presented in the following, addresses all these requirements. Regarding the support for the cost-effective MVP-development, we refine the requirements. The development method shall specifically support the development of so-called *single-feature MVPs*, which are minimum versions of products that allow for gaining validated learnings by experimentation. A single-feature MVP can be considered a "prototype", which only implements the most important function that is required to make informed decisions about the value of a specific function. In more general terms, an MVP can also be seen as the minimum effort required to rapidly learn what the customer wants and needs. MVPs should be developed fast, to establish fast feedback cycles and, hence, foster innovation cycles.

3.2 Overview of the Development Method

Figure 1 provides the birds' eyes perspective on the development method. The development method consists of three basic stages:

1. Feature Requirements Generation and Evaluation (FRGE)
2. Design and User Experience Evaluation (DUXE)
3. Pre-Launch Testing (PLT)

These three stages describe the basic workflow. All three stages must be implemented in a sequence on a per-feature basis. The actual workflow is characterized by a number of artifacts that serve as input and output of the respective stages and that, furthermore, are linked with the specific development methods.

The starting point of the workflow is the *validated problem*. A validated problem can be an issue that has been evaluated and named relevant to customers. To validate a problem, Bosch et al. [Bo+13] suggest asking the following questions: *What is the problem? Who has the problem? Is the problem big enough to make a business out of it?*

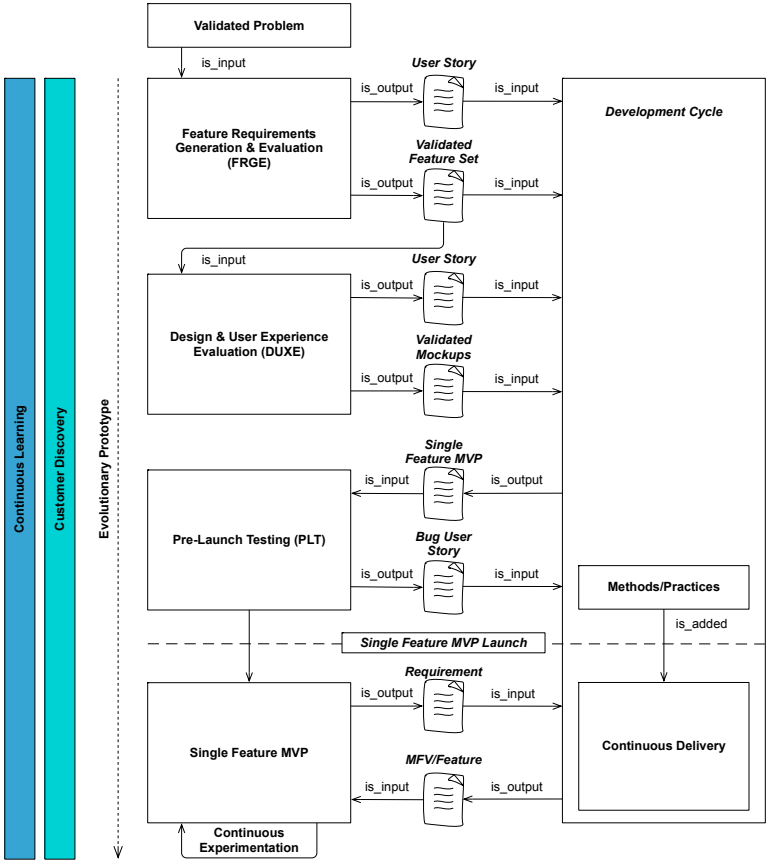

Figure 1 Overview of the hybrid development model for early-stage startups

Once the problem has been validated, the first stage – FRGE – is entered in which the initial feature set is compiled and the single-feature MVP is designed. Since it is the overall goal of the proposed development method, to help early-stage startups develop software fast, right in this stage, first user stories are issued to start the development activities. The second stage – DUXE – aims at ensuring the user experience (UX). It is crucial to start the UX evaluation as early as possible, e.g., with early adopters, to ensure that the features are properly implemented to provide a satisfying experience for the user and usability of the software. To refine the initial requirements and to align them as optimal as possible with the users' goals, experimentation is used. The final step is represented by the stage PLT in which the MVP is tested. Depending on the MVP, the tests range from automated unit test to complex test setups, e.g., installation, mobile usability, and energy consumption.

Key to the entire model is the availability of a continuous software engineering environment, which allows for continuous integration and delivery. Figure 1 also shows that two complementing activities are conducted along the MVP development: continuous learning and customer discovery. In the customer discovery, special emphasis must be put on early adopters (that at best can become "evangelists"). These persons are used to evaluate problems and the respective solution approaches.

3.3 Stage 1: Feature Requirements Generation and Evaluation

The goal of this stage is the collection and validation of the initial features of the software. Especially when aiming for a single-feature MVP, the number of features to be included is very limited and, therefore, the "right" features to be implemented need to be identified. This is especially crucial, since startups – as many other software-producing companies – tend to start coding early.

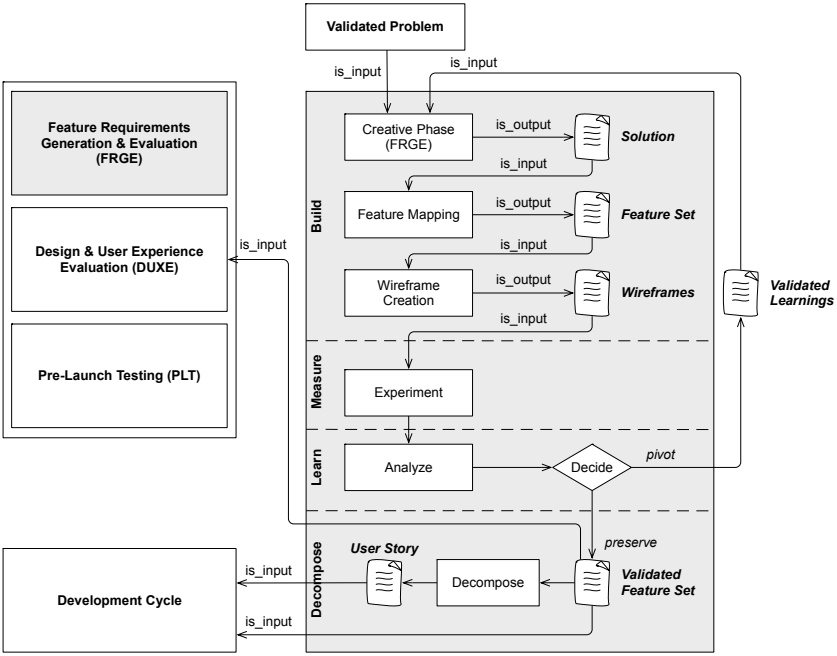

Figure 2 Refinement of the first stage FRGE

To avoid "working for trash" or to consume too many resources, the key requirements need to be identified and designed into the MVP properly. For this, the FRGE-stage is organized according to the Build-Measure-Learn cycle (Figure 2). In this stage, creativity techniques are used to develop an initial solution idea, which is structured using the feature mapping approach. To provide means for an early evaluation, wireframes should be created, which serve as input for the experimentation activities. Validated learnings from the analyses help improve further cycles. In case the feature refinement saturated and the maturity is considered high enough, the validated feature set is decomposed into user stories, which are handed over to the actual software development.

To support the decision-making, i.e., to leave the Build-Measure-Learn cycle, Bosh et al. [Bo+13] suggest two questions: *Is the current set of features sufficient to solve the customer's problem? Are customers willing to test the MVP?*

3.4 Stage 2: Design and User Experience Evaluation

Good user experience is key. Hence, the proposed development method includes a dedicated UX phase as a key element, which is shown in Figure 3.

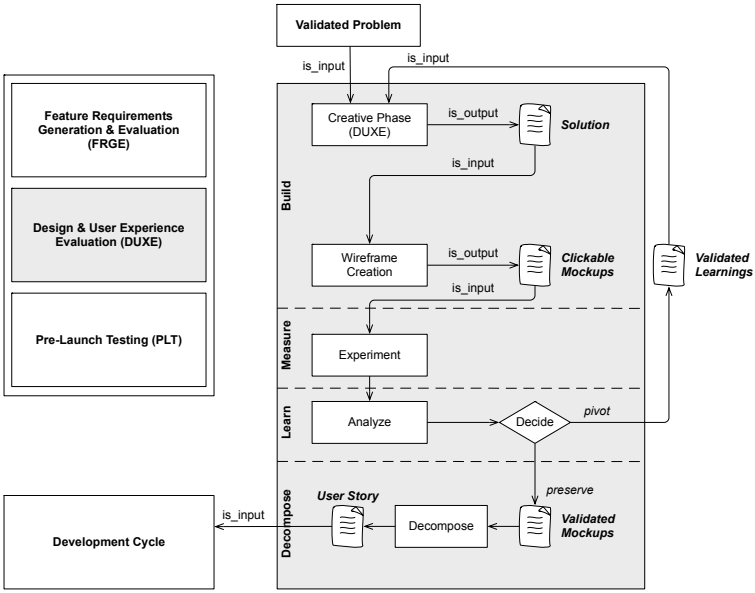

Figure 3 Refinement of the second stage DUXE

Similar to the first stage, the DUXE-stage starts with creativity activities. However, there is no explicit step in which new features are defined. Instead, the solution proposal from the creativity activities results in wireframe designs, which are evaluated. The rest of this stage is organized in the same way as the FRGE-stage. A slight, yet important difference compared to the FRGE-stage is the development of software-based prototypes (Mockups). While a simple, even a paper-based, approach for the wireframing is sufficient for the FRGE-stage, in the DUXE-stage, we recommend using clickable prototypes, since these provide a better means for early adopters and evaluators to test the MVP. A clickable prototype thus helps shortening the feedback cycles and to obtain feedback of better quality, as evaluators use an object that is closer to the desired product, e.g., general look & feel and so forth. Also, the mockups are subject to the experiments

in the Build-Measure-Learn cycle. To support the decision-making, the following two questions should be asked: *Is the user experience sufficient for early adopters? Does the customer fully understand the use of the product?*

Once the results of the analysis stabilized, the feature set together with the prototyped wireframes are handed over to the development. For this, new user stories are created, which are implemented and gradually integrated with the other system components.

3.5 Stage 3: Pre-Launch Testing

Once software is deployed, it remains in the IT-ecosystem. If the quality of the released software is insufficient, quality problems and various risks will occur during the system's lifetime. Therefore, the third essential part of the proposed development method is *quality assurance*.

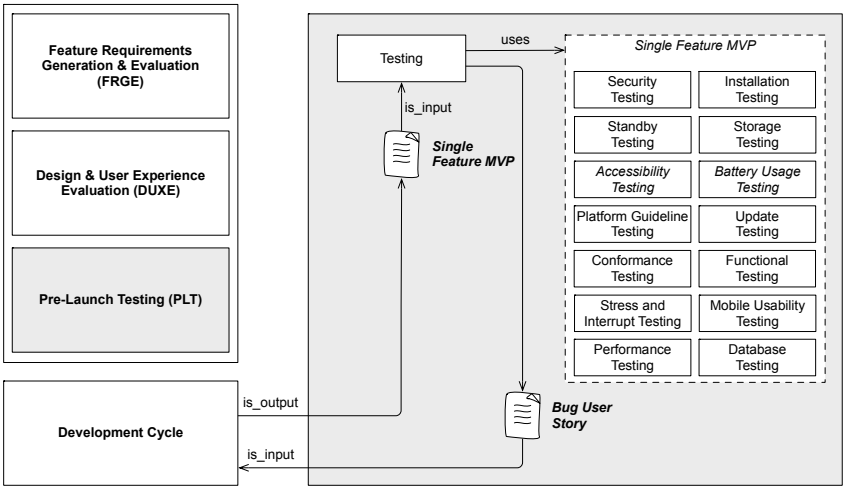

Figure 4 Refinement of the third stage PLT (optional test activities are italic)

Figure 4 shows the refinement of the PLT-stage in which the testing-related activities are located. Once a single-feature MVP has reached a state in which it can be potentially released, it is put into the testing stage. Depending on the actual kind of the product, different testing methods are required, e.g., battery usage testing for smartphone apps or mobile usability testing for progressive Web apps. However, it is important to mention that the test environment must be designed such that a continuous testing and integration is possible. Since all the stages shown above are designed as incremental-iterative processes, (automated) tests should be run as soon as user stories have been implemented. Therefore, a continuous software engineering environment is advantageous, since it is possible to select and implement a user story and, once the user story is implemented, to commit the new code to a repository, which automatically triggers a build agent. However, this also requires a specific setup of development methods and their alignment, which is presented in the following section.

3.6 Detailed Workflow in the Development Cycle

Since the selection of the development methods and practices severely impacts the organizational effort of a development project, but also lays the foundation for a sustainable product development, the selection and combination of relevant methods and practices must be done with care.

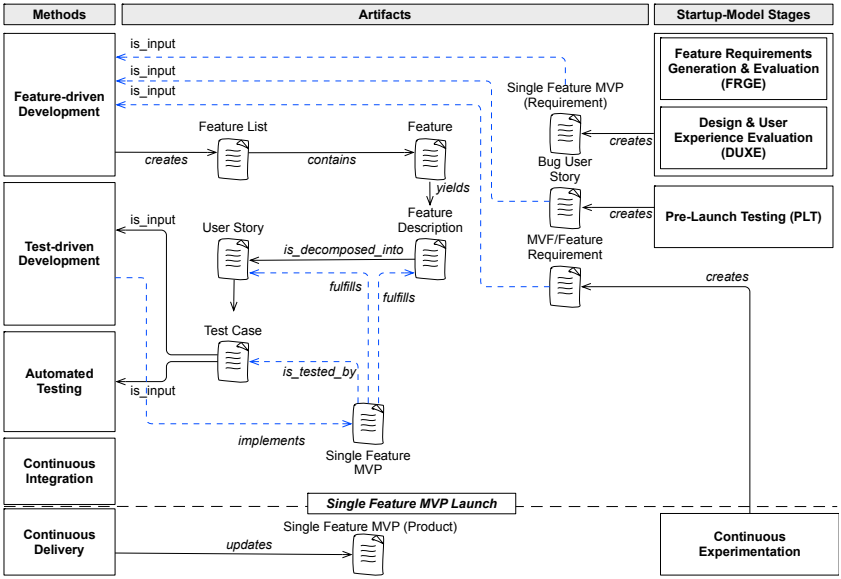

Figure 5 Overview of the startup-model stages, the artifact and the flow of the artifacts in relation to the key development methods

Figure 5 provides an integrated perspective of the development cycle, the artifacts created in the different stages and the combination of development methods and practices utilized in the *Zippr* startup. As Figure 5 shows, the selection of the development methods was driven by their potential for automation. The base method selected is *Feature-driven Development*, since this method already provides a basic idea of the development and organization of feature hierarchies and a per-feature design and development cycle. The actual coding activities are organized using the *Test-driven Development* approach, complemented with automated unit testing. The technical backbone is provided by a continuous integration and delivery infrastructure. Besides the artifacts introduced so far, the continuous experimentation, which is implemented after the single-feature MVP launch, adds the new artifact *Minimal Viable Feature* (MVF), which is created from a hypothesis formulated within the continuous experimentation process.

3.7 Evaluation

The development method presented above is the result from an evaluation performed in an expert review. In total, three experts have been interviewed personally or via Skype. The experts were provided with the initial version of the method alongside with a list of 19 questions (three for the top-level model description, three per stage, and four for the development cycle). The list of questions included questions for the model's consistency, the occurrence of redundancies, the suitability and the completeness of the model.

As a result of the evaluation, the stages FRGE and DUXE have been renamed and their internal structure has been improved (25 comments in total). Furthermore, two more general methods and five extra stage-method assignments have been suggested by the reviewers. In general, the outcome of the expert interview was that the model builds a solid basis for further use and evaluation in practice.

4 Conclusion and Future Work

In this paper, we proposed a lightweight hybrid development method for early-stage startups. The purpose of the method proposal is to provide early-stage startups that potentially lack software design and development expertise with a slim and easy-to-use guideline for organizing the software development activities. Special emphasis was put on the integration of the creativity techniques used to develop and refine ideas to lay the foundation for the software development. This includes procedures to create ideas, to develop feature bundles, wireframes and clickable prototypes and, eventually, to link all these activities to a lightweight method that supports agile and lean software development. Since the method presented is the initial step, a complementing evaluation of the method is subject to future work. This include the critical review of the method and their improvement based on practical learnings. Furthermore, variants of this methods will be developed to provide a more generic framework that supports broader applicability.

References

[Gu15] Guttentag, D.: Airbnb: disruptive innovation and the rise of an informal tourism accommodation sector. In: Current issues in Tourism 18.12 (2015), pp. 1192–1217.

[Wa16] Wasserman, A. I.: Low ceremony processes for short lifecycle projects. In: Managing Software Process Evolution. Springer, 2016, pp. 1–13.

[BD12] Blank, S.; Dorf, B.: The startup owner's manual: The step-by-step guide for building a great company. BookBaby, 2012.

[Gi+14] Giardino C. et al.: What do we know about software development in startups? In: IEEE software 31.5 (2014), pp. 28–32.

[LM16] Lindgren, E.; Münch, J.: Raising the odds of success: the current state of experimentation in product development. In: Information and Software Technology 77 (2016), pp. 80–91.

[Ri11] Ries, E.: The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses. Crown Books, 2011.

[Su00] Sutton, S. M.: The role of process in software start-up. In: IEEE Software 17.4 (2000), pp. 33–39.

[DA16] Duc, A. N.; Abrahamsson, P.: Minimum viable product or multiple facet product? The Role of MVP in software startups. In: International Conference on Agile Software Development. Springer. 2016, pp. 118–130.

[Be+18] Berg V. et al.: Software startup engineering: A systematic mapping study. In: Journal of Systems and Software 144 (2018), pp. 255–274.

[Cr02] Crowne, M.: Why software product startups fail and what to do about it. Evolution of software product development in startup companies. In: IEEE International Engineering Management Conference. Vol. 1. IEEE. 2002, pp. 338–343.

[Fa+14] Fagerholm F. et al.: Building blocks for continuous experimentation. In: Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering. ACM. 2014, pp. 26–35.

[Fa+17] Fagerholm F. et al.: The RIGHT model for continuous experimentation. In: Journal of Systems and Software 123 (2017), pp. 292–305.

[GWA14] Giardino, C.; Wang, X.; Abrahamsson, P.: Why early-stage software startups fail: a behavioral framework. In: International Conference of Software Business. Springer. 2014, pp. 27–41.

[KUG15] Klotins, E.; Unterkalmsteiner, M.; Gorschek, T.: Software engineering knowledge areas in startup companies: a mapping study. In: International Conference of Software Business.

Springer. 2015, pp. 245–257.

[Ku+18] Kuhrmann M. et al.: Hybrid Software Development Approaches in Practice: A European Perspective. In: IEEE Software (2018).

[OAB12] Olsson, H. H.; Alahyari, H.; Bosch, J.: Climbing the 'Stairway to Heaven' – A Multiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software. In: 38th Euromicro Conference on Software Engineering and Advanced Applications. IEEE. 2012, pp. 392– 399.

[OB14] Olsson H. H.; Bosch, J.: The HYPEX model: from opinions to data- driven software development. In: Continuous Software Engineering. Springer, 2014, pp. 155–164.

[Pa+14] Paternoster N. et al.: Software development in startup companies: A systematic mapping study. In: Information and Software Technology 56.10 (2014), pp. 1200–1218.

[TSA19] Tegegne, E. W.; Seppänen, S.; Ahmad, M. O.: Software development methodologies and practices in start-ups. In: IET Software 13.6 (2019), pp. 497–509.

[TSS16] Terho, H.; Suonsyrjä, S.; Systä, K.: The Developers Dilemma: Perfect Product Development or Fast Business Validation? In: International Conference on Product-Focused Software Process Improvement. Springer. 2016, pp. 571–579.

[Bo+13] Bosch, J. et al.: The early stage software startup development model: a framework for operationalizing lean principles in software startups. In: International Conference on Lean Enterprise Software and Systems. Springer. 2013, pp. 1–15.

[Ze+01] Zettel, J., Maurer, F., Münch, J. Wong, L.: LIPE: A Lightweight Process for E-Business Startup Companies Based on Extreme Programming, In: Proceedings of the International Conference Product Focused Software Process Improvement, Springer, 2001.