Date of acceptance Grade

Instructor

Organization of testing in internal startups

Eero-Veikko Laine

Helsinki February 23, 2018

Master's thesis
UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO - HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta - Fakultet – Faculty	Laitos - Institution - Department
Faculty of Science	Department of Computer Science

Tekijä - Författare - Author

Eero-Veikko Laine

Työn nimi - Arbetets titel - Title

Organization of testing in internal startups

Oppiaine - Läroämne - Subject

Computer Science

Työn laji - Arbetets art - Level	Aika - Datum - Month and year	Sivumäärä - Sidoantal - Number of pages
Master's Thesis	February 23, 2018	47 pages + 2 appendix pages

Tiivistelmä - Referat - Abstract

Internal startups are new ventures launched by companies seeking ways for radical innovation. Conceptually part of Lean Startup, they are strongly influenced by independent startup companies and agile software methodology. Internal startups favor a "lean" approach to their organization, usually have minimal resources and are expected to produce results fast. This thesis explores how to organize testing effectively in the difficult conditions internal startups operate in.

We conducted a case study where we interviewed five IT professionals associated with an internal startup in a global IT service and software company. To systematically analyze the material collected in the interviews, we applied thematic synthesis. Our results suggest that the organization of testing in internal startups is affected by the resources provided by the startup's parent company, as well as the demands presented by the company. Our results also suggest that the lean approach favored by internal startups can have a negative effect on testing and product quality. These results are supported by the existing literature on the subject.

ACM Classification:

- **Software and its engineering~Software testing and debugging**
- **Social and professional topics~Project and people management**

Avainsanat – Nyckelord - Keywords

Internal Startup, Lean startup, Software testing

Säilytyspaikka - Förvaringställe - Where deposited

Muita tietoja - Övriga uppgifter - Additional information

Table of contents

1 Introduction

Internal startups are new ventures launched by companies seeking ways for radical innovation [Rie11]. Conceptually part of Lean Startup, they are strongly influenced by independent startup companies and agile software methodology. Like independent startup companies, internal startups favor a "lean" approach to their organization, usually have minimal resources and are expected to produce results fast. Unlike independent startups, internal startups operate as autonomous divisions inside their parent organization.

The startups' lack of effective process management often leads into quality assurance being neglected. Testing is not prioritized in startups, as testing is time-consuming and startups operate under pressure. This thesis explores how to organize testing effectively in the difficult conditions internal startups operate in. We were motivated to study this subject by our own background in testing, as well as our interest in management and software production.

Our research questions are:

RQ1: How is testing organized in internal startup projects?

RQ2: How should testing be organized in internal startup projects?

RQ3: Is the testing of internal startup projects different from the practices companies usually employ and if it is, why?

To provide answers to our research questions, we conducted a case study where we interviewed five IT professionals who were associated with an internal startup in a global IT service and software company. The internal startup team was developing a testing tool for professionals where user interface was not a priority. This made testing of the product demanding.

To systematically analyze the material collected in the interviews, we applied a method for thematic synthesis suggested by Cruzes and Dyba [CrD11], producing a model of the themes emerging from the interviews. The four main themes that emerged were 1) product development, 2) the parent company, 3) management in the project and 4) testing.

Our results suggest that the organization of testing in an internal startup is highly dependent on the resources the startup's parent company allocates to the project. Additionally, the demands presented by the parent company can have a negative effect on the startup's testing. Our results also suggest that internal startups tend to adopt an agile-influenced and lean approach to the organization of testing in favor of traditional management and testing processes and specialized testing roles. We suggest that this approach, while justified, can also have a negative effect on testing and product quality in internal startups. Our results are supported by the existing literature on the subject.

This thesis is organized as follows. In the second chapter, we examine software testing, including traditional and agile testing and the organization of testing activities. In the third chapter, we examine internal startups, their influences, and their role as innovators. In the fourth chapter, we present our methods and discuss our study's validity. In the fifth chapter, we present our results, divided into four main themes. In the sixth chapter we discuss our results with respect to literature and our research questions. Finally, in the seventh chapter, we present our conclusions. Our original interview question set is included in the appendix.

2 Software testing

We explore how to organize the activity of testing, conducted in the context of internal startups. This requires a defined view of testing, which we present in this chapter. We first define the activity of testing, examine the testing process and explain the related core concepts. We then inspect agile software development, a trend which has been reforming testing and software development in general in the recent years. We also inspect DevOps, an agile-related approach that is influential in software testing. Finally, we examine how to organize testing in general and present our view on how agile and "traditional" software development approach testing.

2.1 The software testing process

Koomen and Pol [KoP99] define software testing as "a process of planning, preparation, and measuring aimed at establishing the characteristics of an information system and demonstrating the difference between the actual and required status", emphasizing that the testing process begins before the delivery of the product. The differences between expectations and actual results are referred to as defects. Quality assurance measures may prevent, discover, or even remedy lack of quality; quality is in this case defined as the ability to satisfy stated or implied needs. Testing is a detective quality assurance measure, and its function is to discover deficiencies in quality.

The fundamental test process features, in roughly chronological order: test planning and control, test analysis and design, test implementation and execution, evaluation of test exit criteria and reporting, and test closure activities [SLS11]. In test planning the objectives of testing are defined and the necessary resources for the test process are estimated. Test planning also involves setting up the organizational structure for testing resources. The resulting test plan is revised during the testing process based on information produced by test control, which refers to the monitoring of testing activities. Test planning involves the creation of test strategy, which is a high-level description of the testing objectives and the techniques used to achieve them [SLS11], [WWW17]. As exhaustive testing of the complete system is not possible in practice, the test strategy defines how the testing of different subsystems is prioritized [SLS11]. The prioritization is based on the priority of customer requirements as well as the expected risk and the severity of failures in the subsystem. The priority of customer

requirements is determined by the customer, for whom different functions may vary in importance.

Test analysis includes inspecting requirements and test strategy, and abstract test cases are designed based on this analysis [SLS11]. In test implementation and execution concrete test cases are created and executed based on test preconditions and abstract test cases. A test case contains the defined test preconditions, the inputs (for example, a name and password of the user), and the expected outcome. A test scenario contains several abstract test cases, and concrete test cases may be grouped in suites. In evaluation of exit criteria the test results are compared against set objectives, which may include concrete metrics or measures, such as test coverage. Test coverage refers to the percentage of the code which is covered by testing. Test closure activities include the evaluation of the executed test process and the archiving of the employed testing software.

2.2 The general V-model

Software development lifecycle models such as the Waterfall-model, the general V-model and different "agile" models are used to systematically control the software development process [SLS11]. The lifecycle models typically feature defined phases and design steps, which are considered complete when their results meet the set quality criteria. Usually a model defines a dedicated role for each task to be performed. The lifecycle models aid with planning and allocation of resources such as time, personnel and infrastructure, and the selected lifecycle model usually influences how testing is performed.

The general V-model (Figure 1) is a widely used and influential lifecycle model for software development projects [SLS11]. In the general V-model, testing is equally important to programming and development.

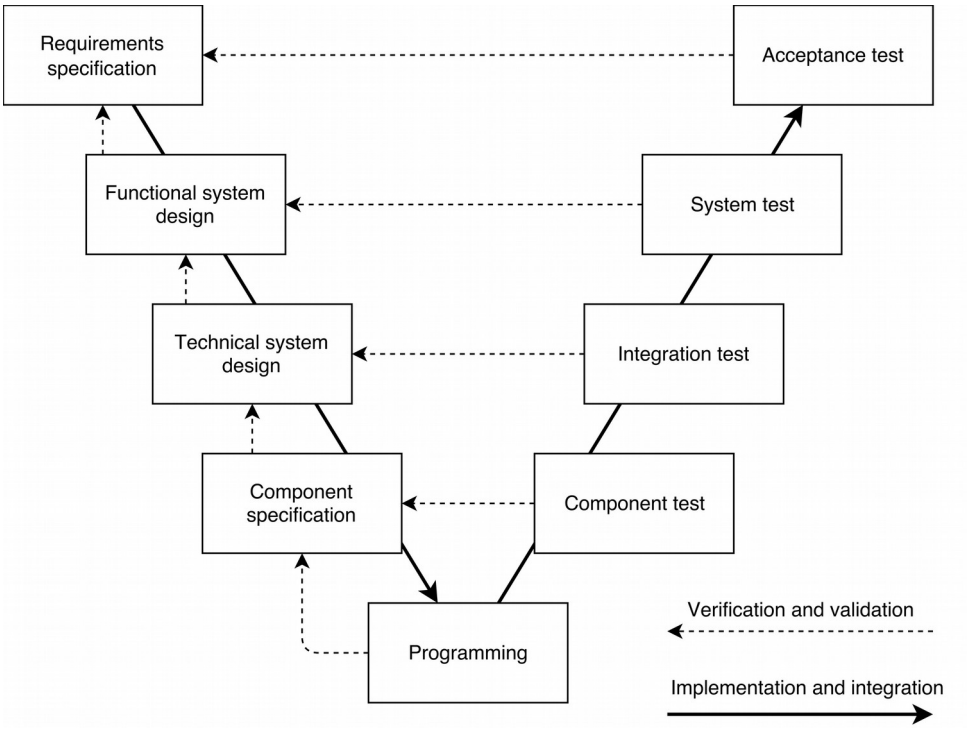

Figure 1: The general V-model [SLS11].

The left branch represents the development process, during which the system is designed, programming being the final step of this process [SLS11]. In requirements specification the needs and requirements of the customer or future system user are gathered, specified and approved. The purpose and the desired characteristics are defined in this step. In functional system design the main functions of the system are designed based on the requirements, and in technical system design the implementation of the system is designed. The technical system design includes the allocation of functions into smaller subsystems, which can be developed independently. The subsystems, including their components, are defined in component specification. In programming step, these components are implemented in some programming language. Test preparation, which includes test planning, control, analysis and design, starts before actual testing and is performed in parallel with the development process.

The right branch represents the integration and testing process, in which different program components are integrated together to form subsystems, and where the functionality of these components is tested [SLS11]. Individual components are tested in the component test step, and the successful interaction of these components is tested in the integration test step. The complete system is tested in the system step against the requirements. In acceptance test the system is tested against requirements from the customer's point of view. Component test is performed by testers together with

developers, and integration and system test may also involve cooperation with developers. Acceptance test is generally performed by testers, the customer or both.

Each test step of the V-model features validation and verification testing [SLS11]. Validation refers to testing against the requirements, while verification ensures that the outcome of a development phase has been achieved according to the phase's documented specifications. In addition, four different types of testing can be distinguished: functional and nonfunctional testing, testing of software structure and change-related testing. Functional testing refers to testing which verifies the system's input/output behavior against specifications. Black box methods are generally used; these methods test the system only through a user interface, without inspecting the source code. Structural techniques, also called white box testing, use information about the system's internal structures, such as source code or architecture. In nonfunctional testing, the reliability, usability and efficiency of the system are tested. Change-related tests include regression tests, which are retests of previously tested programs performed after their modification to ensure there have been no changes to the program's functionality. The tests that only cover the main functionality of the system under test are called smoke tests [WWW17].

2.3 Test automation

Automated test execution, or test automation, is a popular strategy to minimize the cost for software testing, and it holds an important position in modern software development [WES12]. Karhu et al. [KRT09] define test automation as the "automation of software testing activities including the development and execution of test scripts, verification of testing requirements, and the use of automated test tools". Software build process, unit and component tests, web service testing, GUI testing, load tests and test data creation can all be automated [CrG09]. There is also a significant amount of research conducted on automatic test case generation, but the proposed techniques have not been applicable to real software systems [ABC13]. Usability testing and exploratory testing rely too much on human component to be successfully automated in the foreseeable future [CrG09].

Manual testing is time-consuming, and test automation increases efficiency, especially in regression testing [KRT09]. Test automation can be utilized to increase test coverage, but human involvement is needed in the selection of test cases. As test

automation is not always cost effective, manual testing is still a viable solution in some cases. Test automation is seen to consume resources heavily, as the complexity of automation tests grows over time, and the tests take progressively longer to maintain and update [DeS13].

2.4 Agile software development and agile testing

Agile software development encompasses multiple different approaches to software development which share many of the core values of the Agile Manifesto [ElS16]. These approaches include Extreme Programming, Scrum, the Dynamic Systems Development Method, Crystal, agile project management, feature-driven development, and lean software development. In general, Agile approaches stress both customers' and developers' needs in software development. Iterative development cycles, frequent communication with customers, constant adaptation and responsiveness to change are common themes. According to Dingsøyr et al. [DNB12], "agility entails ability to rapidly and flexibly create and respond to change in the business and technical domains". Lightness, speed of response and "leanness", i.e. having minimal formal processes, are a significant theme in agile software development. The influence of Agile Manifesto has been unprecedented in software development, and this is reflected in the vast number of different agile methods.

Crispin and Gregory [CrG09] emphasize the variety of agile testing, but present iterative development cycles and cross-functional development teams as the agile norm. Development teams include everyone involved in delivering code, and individual members of the team are allowed to perform all tasks of the team. Specialized roles are discouraged, even if the team must have all the expertise the project requires, involving programming, architecture, system administration, and quality assurance. Customer representatives can be included in the team, and the team can be restructured as needed. Agile software development favors so-called whole-team approach, where all team members are responsible for the quality of the software. In practice this can mean developers taking responsibility for all kinds of testing tasks.

Iterative development cycles, also called iterations, are short periods of software development [CrG09],[Dru14]. A typical length of an iteration is two weeks. Objectives are developed for each iteration and reviewed at the end of the iteration [Dru14]. The functional objectives of the iteration include the creation of some functionality, the

testing of what has been developed during the iteration and the creation of documentation. Additional objectives involve quality, team satisfaction and staying on schedule.

An agile team may release software each iteration (Figure 2) [CrG09]. In this case, the last one or two days of each iteration are reserved for user acceptance testing, training, bug fixing and other finishing activities. Some teams release every few iterations, and in these cases the iteration before the release may be reserved for the finishing activities. However, testing is still a continuous part of the development process.

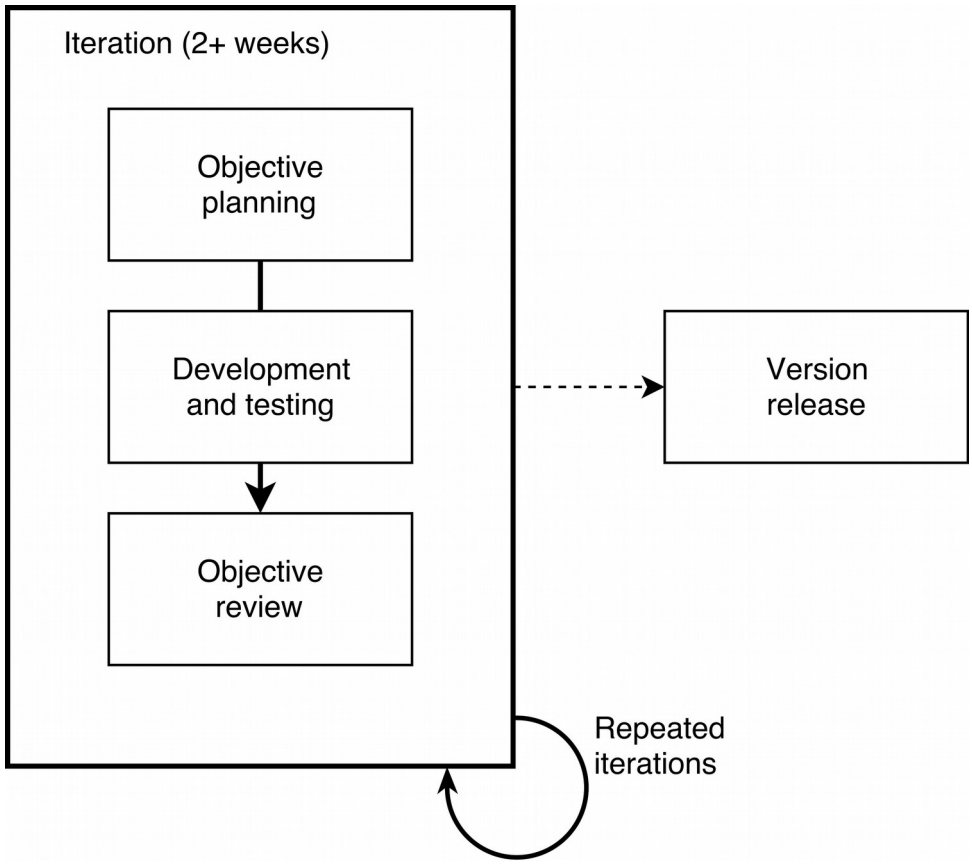

Figure 2: The iterative process model.

We suggest that the V-model and the iterative model are markedly different approaches to software development. In the V-model, the lifecycle of the development process is divided into distinct phases that apply to the whole software product and may involve different teams. In the iterative model, the development process consists of iterations, during which the whole development team develops, tests and even releases working software with fully finished functionality; in this model, the whole V-model lifecycle is condensed into a single iteration.

Agile team members can employ "stories" in planning and prioritizing [CrG09]. A (user) story is a brief description of desired functionality written from the perspective of the user. A typical story reads like the following: "As a shopper on our site, I want to delete items out of my shopping cart so I don't purchase extra items that I decide I don't want." [CrG09]. Tests performed during the iterations are not based on the requirements document of the overall system, but short-term requirements related to each story [CrG09]. These requirements are often created collaboratively by a business or domain expert and a development team member such as a tester. The requirements are supported by detailed, functional test cases, ideally based on examples provided by business experts.

To find defects outside the scope of the defined test cases, testers perform manual exploratory testing. Execution and automation of test cases may proceed in parallel to implementation of each story. The story is considered finished when the tests that demonstrate minimum functionality are complete. One way to measure completion is Definition of Done [SAN17]. Definition of Done is a popular, Scrum-based agile concept, which consists of a set of criteria which define when a component of a product is ready for release.

Test-driven development is a core practice of agile testing [CrG09]. In test-driven development, the programmer first writes a test to test a part of some functionality and then writes the code that makes the test pass. The purpose of test-driven development is to increase code quality. While agile testing is known for its emphasis on test-driven development and acceptance testing, load testing, security testing, performance testing and usability testing, among other types of testing, are just as vital in agile projects as they are in traditional ones. Overall, agile testing is characterized by short time frames, quick feedback and shared responsibility for quality.

2.5 DevOps

DevOps is a new, agile-based approach to software development which has a significant impact on the entire software industry [EGH16]. In DevOps, software development is integrated with quality assurance, infrastructure operations, and business processes. End-to-end automation, which comprises the whole software development workflow from programming to delivery, is a critical goal. DevOps teams are cross-functional and responsible for the entire delivery process.

Continuous integration and continuous delivery are a critical part of DevOps [EGH16]. In continuous integration developers test their code by running private builds before committing their code to the version control repository, which is done at least once a day [MSB16]. Dedicated integration builds are run several times a day against automated tests. The resulting build must be functionally testable, and all tests must pass. Fixing broken builds is a high priority.

Continuous delivery is the ability to release software whenever the project team wants to do so [NeS13]. The purpose of continuous delivery is to make the latest features and upgrades available to users as fast as possible. To achieve continuous delivery, DevOps approach expands automation, including test automation, from the application level to the infrastructure [EGH16].

2.6 The organization of software testing

The organization of software testing involves the allocation of testing tasks to testing and other personnel [SLS11]. Testing is ideally performed concurrently with development activities, and the simplest option is to have the developer test her own code. However, this option has several downsides. Developers tend to be blind to their errors, and independent testers find different defects than developers. Developers also make implicit assumptions of the system under development during the specification and implementation of the system, and independent testers can verify these assumptions. Kasurinen et al. [KTS09] argue dedicated testing staff can do testing work more efficiently compared to developers. Nevertheless, many defects are found by people who are not testing specialists, such as developers and sales personnel [MII12].

Different models for independent testing can be identified [SLS11]. One option is to have the developers test each other's code. There can be dedicated testers within the development team, or dedicated test teams within the project team, working alongside the development team. There can also be testing specialists for specific testing tasks such as performance or usability testing. A separate organization, such as the company's testing department, may be responsible for testing. Outsourcing testing to another organization is also a valid option, but domain expertise, management and vision is required of the client organization [CrJ02].

Different roles for testing personnel include test manager, test designer, test

administrator and tester [SLS11]. Test manager has expertise in software testing as well as quality, project and personnel management. Test manager may be responsible for writing the test strategy, monitoring testing work and selecting suitable metrics and testing tools for the project. Test designer is typically responsible for reviewing user requirements and creating test specifications based on them. Test administrator cooperates with system administration and network management to set up and operate the test environment. Dedicated testers execute tests and evaluate and document results. In addition there can be dedicated test automators. The testing team may also need the assistance of non-testing personnel such as domain specialists and database administrators.

Craig and Jaskiel [CrJ02] argue that traditional, separate testing units influence software quality positively. In a dedicated testing unit testers focus on testing alone, and their expertise reflects this. Craig and Jaskiel et al. argue testers in a dedicated testing unit are not pressured to release software with quality issues like the testers in integrated teams, and their separation from the development allows them to have a more objective outlook on the product. According to Craig and Jaskiel, a testing unit manager should be organizationally on the same level with the development team manager (Figure 3). However, the independent team model may lead the developers to think that quality is the testing team's responsibility alone. If developers fear testers slow their progress, the testers are not allowed to perform testing as early as possible. Kasurinen et al. [KTS09] also recommend a clear separation of testing resources from other project resources to preserve them.

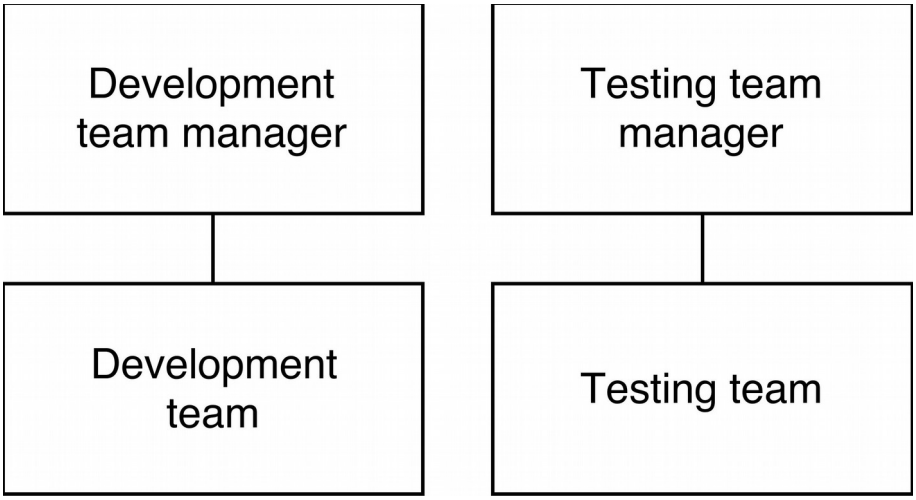

Figure 3: The testing organization model of Craig and Jaskiel.

Crispin and Gregory [CrG09] recommend that testers are integrated in the agile development team (Figure 4). All personnel in the agile development team are expected to commit to quality assurance, but testers can take the role of a customer advocate and influence the development to prioritize quality. When testers are colocated with developers, communication and team-building become easier. However, Crispin and Gregory imply testers should not give up their role as a tester, but be advocates for quality assurance and work together with developers to improve product quality.

The agile team members should have an easy access to all team members and easy visibility of all progress charts [CrG09]. The team manager or team leader must also ensure that testers are able to receive technical and other help inside the team. Testers also need time and training to adjust to working in an agile project. An experienced agile team member can coach new agile team members or members of an existing team that is transitioning to agile software development. The agile model for organizing testing has been a popular approach in the software industry [DeS13].

Different phases of testing can benefit from different approaches: testing of separate components is ideally performed in co-operation with developers, so a model where dedicated testers work in a development team is a good fit for this case [SLS11]. The same principle applies to a case where components developed within the same

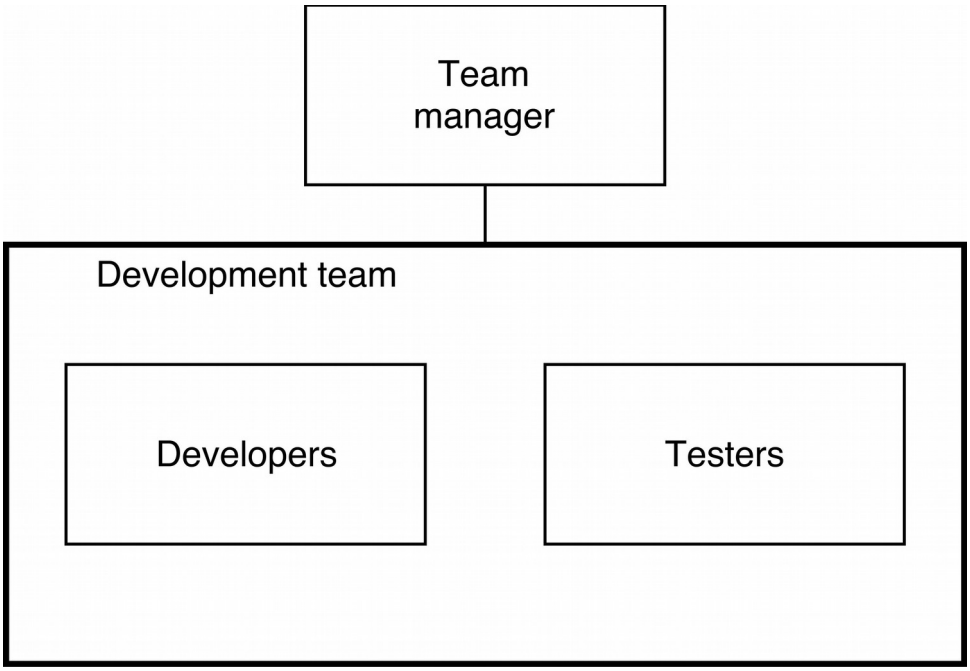

Figure 4: The testing organization model of Crispin and Gregory.

development team are integrated. However, when components of several development teams are integrated, a separate testing team or department may detect defects more effectively because of their "outsider" perspective. In system testing, where testing is performed from the customer's point of view, independence from the development is even more important. Domain knowledge is highly relevant in testing: validation by domain experts such as customers and end users may be valued more than any technical verification [MII12].

2.7 Social factors in software testing

Social factors such as the company culture have an influence on the testing process [MII12]. Software defects are prioritized very differently by people in different positions. In one case study, defects were prioritized higher by the company's sales and consulting personnel, who had a personal stake in the product. However, this did not improve the probability of a defect to get fixed. Defects reported by developers were fixed most often, while the defects reported by specialized testers were fixed least often. Defects assigned to developers in different teams than the defect reporter are less likely to be fixed [GZN10].

Among software developers, testing is seen as a secondary position [CBG04],[DeS13].

The testers' secondary status makes their job more difficult and time-consuming [CBG04]. The support of the management is crucial: if the management does not support the testers, the developers do not respect the testers' opinion. Developers often take criticism of their code personally, which leads into conflicts between testers and developers. Testers tend to emphasize usability in software, while developers value novel technical solutions which may ultimately be incompatible with the requirements. Another source of conflict between testers and developers is the allocation of time between development and testing, as time spent on development is usually taken from testing. To alleviate the conflicts between developers and testers, active leadership, time management and team building are required. Co-location of testers and developers is also an effective way to address conflict between them.

2.8 Discussion

In this chapter, we have attempted to present an overview of software testing, emphasizing the aspects most relevant to our thesis. In this particular subchapter we wish to summarize, review and discuss the views found in literature from our own point of view.

Based on literature, we argue agile software development takes a different approach to the organization of testing compared to pre-agile, or "traditional" software development. Agile software development is based more on trust than control: all members of an agile development team are expected to emphasize program quality and work to maintain it without any outside pressure. Agile team members are specifically coached to adopt a more collaborative, quality-oriented mindset, while in traditional software development the testers are responsible for controlling the program quality.

In traditional software development, the developers cannot be held responsible for quality assurance, as they are considered to be compromised by the pressure to stay on schedule. As testing in the traditional model generally follows development, time used for development is frequently taken from testing, and testers have to defend the importance of their work. The difficult position of testing in the software industry has led authors [CrJ02],[KTS09] to recommend protective measures such as separating testing resources entirely from the development resources, and keeping testing and development teams equal in the organization hierarchy.

The formal separation of testing resources from development resources and rigidly defined tester roles and organization structures are meant to ensure that no forms of testing are overlooked, that testing is protected from the pressure to release, and that testers can develop their expertise without other commitments. However, in a successful agile development team software quality is supposed to be a shared, internalized value of all team members. If the development team is committed to quality, the need to protect testing activities or testers' expertise with formal organization structures is reduced.

In traditional testing, requirements, considered to accurately model the customer's wishes, are the foundation for all testing activities. However, in agile software development the requirements are fluid and software quality is measured with customer feedback throughout the project. The entire agile development team may work directly with customer representatives, and all team members, irrespective of their specific expertise, are held responsible for taking customer feedback into account.

In addition, we suggest authors discussing the agile approach use different terminology compared to other authors. While agile-oriented literature acknowledges different types of testing such as integration or acceptance testing, there are no mentions of planning aids such as test strategy or test planning, and the agile testing solutions are presented as flexible guidelines rather than systematic processes. These terminological differences sometimes make direct comparisons between agile and more traditional approaches difficult.

3 Internal startups

Testing and the organization of testing activities are informed by the context where the testing is performed. In this chapter we inspect independent startup companies and examine how the Lean Startup approach applies their techniques in other contexts. We then inspect how established companies seek to innovate by creating internal startup projects. Finally, based on literature, we briefly discuss internal startups and how they organize their testing activities.

3.1 Startup companies

Startups can be defined loosely as "newly created companies with no operating history and fast in producing cutting-edge technologies" [PGU14]. Startup companies are new, small and inexperienced [Sut00]. They have limited resources, and the first resources invested in the company are usually spent for promoting a product. A startup company is sensitive to multiple influences such as investors, customers, partners and competitors, and are often established to develop technology that is seen as innovative or "cutting edge". While these characteristics apply to established companies as well, they define startup companies to an extreme degree.

In the 2010's startups are also defined by fast growth, time pressure, third party dependency, single-product focus and flat organizational structures [PGU14]. Startups are forced to release their product fast and to work under constant pressure, in part because of investors' requests; because of their limited resources, startups are forced to rely on external solutions such as open-source software and outsourcing to build their product; and startups are centered on founders, while everyone in the company has big responsibilities. Overall, the most reported characteristics of startups in literature are lack of resources, uncertain conditions, flexibility, innovation and fast growth.

Time pressure and lack of resources lead to a loose organizational structure and lack of traditional project management and documentation in startups [PGU14]. When organizational practices are flexible, startup members must be empowered to act independently and influence the project. The startup's goals should be planned for short term. The absence of structure makes activities such as knowledge management or team coordination more difficult, especially when the startup grows. As with testers and developers [CrG09], the co-location of team members facilitates informal

communication and the coordination of business and technology strategies [PGU14].

Because startups lack effective process management, quality assurance is often neglected [PGU14]. Software testing is time-consuming and not prioritized, and test automation is not yet accessible enough to be widely used. Usability tests, user interface testing and early and frequent user validation are reported to be useful in startups. However, the core functionalities of the software should be the priority of the startup team. To maintain the focus on core functionalities, even outsourcing of the testing effort has been suggested. Startups can also use A/B testing to test their product by conducting user experiments [LiM16]. In A/B testing, users of a software service are randomly divided in two groups, and each group uses one variant of the service [HyK14]. Feedback from the users is collected, and one of the variants is discarded based on the results [HyK14],[Tam14].

3.2 Innovation and Lean Startup

Innovation is one key aspect of startups [PGU14]. Large companies seek ways for radical innovation, which allows them to transform an existing market or to create a new one [EWA15]. According to Crossan and Apaydin [CrA10], innovation is "production or adoption, assimilation, and exploitation of a value-added novelty in economic and social spheres; renewal and enlargement of products, services, and markets; development of new methods of production; and establishment of new management systems. It is both a process and an outcome." Value-added novelty in this case refers to the intended benefits of new practices.

Companies struggle to innovate partly because they have invested much resources in the existing technology and market [EWA15]. A company's capability to innovate is also hindered by the size and complexity of its operations. The company may focus too much on its current position in the market and satisfying current customers rather than exploring new ventures. For innovating inside corporations, Edison et al. [EWA15] suggest Corporate Entrepreneurship, a model where a company innovates continuously to explore new products and business domains.

One approach to entrepreneurship is Lean Startup, an agile business methodology developed by Eric Ries [Rie11]. By Ries' definition, a startup is "a human institution designed to create a new product or service under conditions of extreme uncertainty".

According to Ries, the fundamental activity of a startup is the Build-Measure-Learn loop, where ideas are turned into products, customer feedback is measured and the product is improved by analyzing the results. In Lean Startup, The Build phase of the Build-Measure-Learn loop produces a functional Minimum Viable Product (MVP), which is developed as quickly as possible. The MVP is then evaluated by potential customers and possibly even sold to them. The MVP is developed further after the Measure and Learn phases of the loop. However, Ries stresses that startups are based around the entrepreneur's stable vision, a core business idea which the experiments must serve. In addition, the startup's progress must be measured, milestones must be set and work must be prioritized. Lean Startup has been a popular approach within startup community, but its methods are considered difficult to implement because of their vagueness [BLB13].

3.3 The role of internal startups

According to Ries [Rie11], anyone may be an "entrepreneur", i.e. a startup facilitator. While the term "startup" has originally applied specifically to independent companies [PGU14], Ries refers to "internal startups", which are autonomous divisions inside a parent organization. Ries argues startup teams need "complete autonomy" to develop and market new products, but "within their limited mandate", implying the limits to the internal team's autonomy must be clearly set.

Internal startups are new ventures which allow the companies to take risks and innovate proactively, even if limited by their available resources [EWA15]. If company management recognizes a potent market segment, it can initiate an internal startup to target it. An internal startup always operates within the boundaries set by the parent company's management. Compared to independent startup companies, the advantages of internal startups include access to resources of the parent company, the established brand and reputation of the company and the specific market of the company. However, the parent company must commit resources to the internal startup, which makes the venture costly. The company's brand or reputation may also be harmed by the startup's actions.

Edison et al. [EWA15] argue entrepreneurship cannot be successful in a large, bureaucratic corporation. An internal startup requires autonomy and freedom to innovate without following existing procedures. Ries [Rie11] requires that the internal

startup is able to build and ship functional products. Ries also recommends that internal startups have full-time representation from every department involved with the product. However, making the parent company commit to an internal startup is difficult [EWA15].

The parent company can be considered a "customer", whose needs the startup team must serve to secure the resources they need [EWA15]. An internal startup needs a champion, a person in a senior position who can protect the project from outside pressure. Edison et al. refer to an internal startup project which was receiving complaints from other teams in the company for their disregard of standard company procedure. The startup was not protected from the changes in the corporate strategy of the company, and eventually the whole project was terminated.

3.4 Discussion

Based on literature, we suggest that the parent company's role is critical in the operation of an internal startup. The parent company is responsible for providing all resources for the startup, and these resources determine the startup's scale. How the company defines the internal startup's "complete autonomy within their limited mandate" is critical, as the mandate's limits effectively define the startup's degree of autonomy. Ultimately, all autonomy and resources of the startup can be taken away, and the whole project can be terminated. This lack of security and resources can be compared to the uncertain conditions independent startup companies must endure.

As internal startups are under pressure to prove the startup's worth to the parent company leadership, we suggest even incomplete features may be prioritized over quality assurance. While quality assurance, including acceptance testing, is crucial from the user's point of view, we suggest answering to the parent company's needs is even more relevant to the internal startup. In this case product quality, however defined, is prioritized only if the parent company representatives can both verify and appreciate it. We suggest the users' needs, as well as quality assurance, are prioritized only when the internal startup team's position is already secure enough.

Agile software development favors an informal organizational structure and emphasizes flexible rather than defined roles inside the development team. As literature on internal startups emphasizes the "leanness" and informal structures of

the agile approach, we suggest the internal startup is likely to adopt an informal organization, especially if the startup is under a threat of cancellation. Because of the parent company pressure, the informal legacy of agile software development, and the general disregard of quality assurance in independent startup companies, we suggest internal startups are also likely to adopt an informal, ad-hoc approach to the organization of testing specifically.

4 Methods

4.1 Study design

Our research objective was to explore how testing should be organized in internal startups. While the solution is dependent on the stakeholders' priorities, we can still examine the effects of different practices in the particular context of the internal startup. To understand the specific challenges of this context, we seek to understand how testing is currently organized in internal startups, and what factors contribute to the current situation. Finally, we examine if the testing practices in internal startups are different compared to practices companies usually employ.

Our research questions were:

RQ1: How is testing organized in internal startup projects?

RQ2: How should testing be organized in internal startup projects?

RQ3: Is the testing of internal startup projects different from the practices companies usually employ and if it is, why?

We selected case study as our main research method. Case study can be defined as "an intensive study of a single unit for the purpose of understanding a larger class of (similar) units" [Ger04]. Case studies examine a single case in depth in a systematic way, helping the researcher to understand specific causal factors and potential subjects of future research [VST09]. They can be used for both generating and testing hypotheses. In our case, we studied a single internal startup project to understand internal startup projects in general. The answers we provide to our research questions should be interpreted mainly as hypotheses for further research.

To collect data about our case, we conducted semistructured interviews with the project members. Semistructured interviews are more conversational compared to structured interviews, which only use predefined questions without variation [Lea14]. Semistructured interviews use preset questions, but also allow the interviewer to continue dialogue freely on the subjects he considers important. The purpose of the semistructured interview is to obtain descriptions of the interviewees' circumstances from their point of view; these descriptions help the researcher to interpret the phenomena which are being examined. Qualitative research interviews are often ambiguous and incoher-

ent, and there is a risk that the researcher ignores this ambiguity to present coherent themes. The researcher should be open to multiple interpretations and complexity regarding the interviewee's comments instead of looking for a single, coherent view.

4.2 The case project

We aimed to answer our research questions by conducting a case study on an internal startup software project of a global IT service and software company. The company has over ten thousand employees globally and its headquarters is located in Finland. The author of this paper was employed by the company when the study was conducted, and had previously worked for the project inspected in this paper. The case study was not commissioned by the company and the author of this paper was not financially compensated for conducting it.

The case project had its roots in a global innovation program organized by the parent company in 2015. The two key persons in the project had a concept of an easy-to-use test automation and continuous integration tool for developers and testers which would also utilize several existing test automation and continuous integration tools. The product, which we refer to as "the case product", utilizes container technologies such as Docker[1] to continuously test and deploy the software.

The key persons were provided technical and personnel resources by the parent company to launch a small-scale internal startup project. The project has had no official project manager. The majority of the personnel working in the project have been directly developing features, but the roles in the project have been flexible. At the maximum, five persons were working in the project at any given time, a majority of them part-time. Less than ten persons have worked in the project during its lifetime, and only two of them continuously. At the time of writing this thesis, the project was ongoing.

To collect data about the project, semi-structured individual interviews were used. The potential interviewees were selected partly based on suggestions by the unofficial project manager of the project, who was our original contact person and knew the objective of the study. The selected persons had all been involved in the project at some point. All five persons who we approached agreed for an interview.

4.3 Data collection

The interviews were conducted in 2016. We interviewed two key project team members, two other team members who had since left the project, and a tester from the same company who was familiar with the project. Four interviewees were experienced testing professionals while one interviewee was an experienced developer.

Three interviews were conducted in a parent company office and two interviews were conducted via internet calls. The interviews were conducted in Finnish. All interviews were recorded with the interviewees' permission. The interviews took approximately 50 minutes except one interview which took 30 minutes. The 30 minute interview was shorter than the other interviews because the interviewee had not worked in the project and had limited information of it. The interviews proceeded according to our interview structure, with additional questions asked as needed. The answers of the interviewee were not often concise or immediately clear, and the interviewee was allowed to talk freely of his experiences and opinions to provide context for his answers.

The interviews were semi-structural and individual. We presumed that an hour-long interview would adequately cover all the subjects we wanted to discuss. A longer interview would have also been more impractical and demanding for the interviewees. Based on literature, we created a set of questions to present for the interviewees. The questions in the question set were divided to Interviewee background, Case project bacground, Parent company support, Testing in the case project, Best practices in testing and Closing remarks. The English translation of the question set is presented in the Appendix 1: The Interview Question Set.

The question set was supposed to take approximately an hour for an interviewee to answer. Because of the semi-structured format of the interviews, the questions asked during an interview could be omitted or complemented depending on the interviewee's answers. If the answers provoked more questions, these questions were asked immediately. If there was no time for the questions deemed least significant, this was not considered a failure. Our objective was to investigate the big themes related to the research questions, not to elicit simple, narrow answers to predefined questions.

4.4 Analysis

Based on the recordings, a transcription was produced manually of each interview. To analyze the transcriptions systematically, we used a model for thematic synthesis provided by Cruzes and Dyba [CrD11]. Thematic synthesis is a systematic approach where recurring themes in the primary text are identified and analyzed. The objective of thematic synthesis is to examine the primary text in depth and produce conceptual innovations. This is achieved by coding segments of the primary text and synthesizing those codes into themes. The relationships of codes and themes are visualized in a mindmap-like model, which is intended to clearly illustrate the connections between different concepts. Cruzes and Dyba refer to the model as a "conceptual representation of observed phenomena". The steps of thematic synthesis suggested by Cruzes and Dyba are, as numbered by us:

1. Data extraction from the primary studies

2. The systematic identification and coding of concepts, categories, findings and results in the primary text

3. The grouping of codes into themes, subthemes and higher order themes

4. The exploration of relationships between themes and the creation of a model of higher-order themes

5. The assessment of the reliability of the interpretations made in previous steps

The following subchapters present our implementation of thematic synthesis.

4.4.1 The coding process

In the case of our study, the primary texts were the transcriptions of the interviews. Our method of data extraction in step 1 was the interview process. In step 2, segments of the primary text are labelled with descriptive tags, also called codes. The aim of the coding is to "quantify content in a systematic and reliable manner". In our study, we had to decide how to divide the transcriptions to codable segments, and how to formulate the coding system used to code the segments. The segment length can vary based on the method used in data extraction and text volume.

Our transcriptions were split into the interviewer's and interviewee's comments, but the interviewer's brief affirmations, such as 'yes' or 'I agree', were notated inside the

interviewee's comments in parentheses to form continuous replies of reasonable length. More significant interruptions by the interviewer, such as "But how about test automation?", were treated as their own comments, even if the interviewee's reply would still be going on. Our logic here was to treat our own comments as questions and the interviewee's comments as replies to them, even if this was not always strictly the case.

We used individual replies directly as segments and while this was a relatively logical method of segmentation, the replies varied significantly in length. A long, rambling reply could feature multiple themes and topics, and we still had to use a single code to tag it. However, we argue our segmentation was sufficiently detailed to reveal the most relevant messages the interviewees were conveying, as over five hundred segments were assigned a code during the process.

Cruzes and Dyba present three approaches to coding, which are the deductive, the inductive and the integrated approach. In the deductive approach the codes are designed before using, while in the inductive approach the codes are developed inductively during the coding process. In the integrated approach a start list of codes is used, and more codes are added to the list during the process. Our inductive-approach-based solution was to initially give each key statement expressed in the interviews an individual code such as "Testing is the most essential component of Continuous Development", even though this produced hundreds of detailed codes. Typically a detailed code would be used only to code one or two segments. We applied this approach on the four 50-minute interviews. After this process, we were confident enough to abstract the detailed codes down into 46 high-level codes.

The high-level codes were used to tag the segments of the 30-minute interview transcription and retag the other interview transcriptions. As an example of this translation of detailed codes to high-level codes, the high-level code "The case project schedule" was composed of five detailed codes: "Months spent with the case project", "Time the case project has taken", "The case project launch took a lot of time", "The case project launch period was mostly conceptual" and "The case product technologies were defined in the launch phase".

4.4.2 Forming the themes

The purpose of themes, according to Cruzes and Dyba, is to "reduce larger amounts of

codes into a smaller number of analytic units, and help the researcher elaborate a cognitive map". Cruzes and Dyba propose a granular hierarchy of themes composed of codes, and higher-order themes composed of themes. In step 3, we created four themes using the high-level codes as a starting point, and divided the four themes into few coherent subthemes. As there were only four fairly indepent themes, we did not form any higher-order themes. In step 4, we explored the strongest relationships between the subthemes by mapping connections between them, which produced the simple thematic model presented in chapter 5.

4.4.3 Evaluation

In step 5 we evaluated our process of thematic synthesis.

The coding process was very work-intensive, and we found the segmentation, creation of codes and coding of segments very subjective. While the coding process efficiently guided us to analyze the material thoroughly, our model did not naturally emerge from the codes, but had to be formulated through writing, feedback, analysis and re-writing. The codes used to label the text segments provided only a starting point, and they were not eventually used in our final thematic model. However, we trust a more experienced practitioner can achieve good results by strictly following the process proposed by Cruzes and Dyba.

4.5 Threats to validity

We examine the threats to the internal, external, and construct validity of our research. Internal validity refers to the validity of the claims about causal factors in the study, while construct validity refers to the validity of research procedures and the set of measures used in the study [GRW08]. External validity refers to the generalizability of the results of the study.

4.5.1 Internal validity

While we draw conclusions based on literature and the results of our study, these conclusions must be considered as suggestions, and we do not aim to directly prove causality of any factors with this study.

The analysis of the recorded material is subject to bias and reflects our chosen point of view. This bias poses the most significant threat to validity. We utilized a rigorous

coding process to avoid overlooking important themes, but the creation of the codes, the coding of segments, and the formation of themes and subthemes and the connections between the subthemes all proved to be very dependent on our own point of view. We also acknowlegde a risk of attempting to form too coherent narratives from the interview material, ignoring some of the ambiguity and incoherence of the interviewees' comments.

4.5.2 Construct validity

The vagueness of core concepts such as organization of testing and agile software development is a threat to the validity of our study. While the allocation of testing tasks to personnel is for us the core activity in the organization of testing, it was often difficult to conceptually separate activities such as planning, strategy, organization and the actual execution of testing. Agile software development encompasses different methodologies, and features interpreted central to agile methodology by us may be seen as peripheral by other researchers.

The selection process of the interviewees was based on suggestions made by a key startup team member. While this selection may have been biased, it helped us to connect with prominent team members, which was vital for our study. All interviewees gave a detailed, critical and believable account of the case project, but their perspective was limited to the one of developers and testers. As no other prominent team members apart from the interviewees were mentioned during the interviews, we believe activity in the team was well covered by the interviews. However, the point of view of managers, coaches and sales personnel is missing from the study. Especially the rationale behind the parent company's actions could have been explored better by interviewing management personnel.

Despite the focus on productization in Lean Startup, we were unable to provide a clear, comprehensive overview of the case product. As other products were used in connection with the case product, it was often difficult for us to determine the limits of the product's capabilities. While the case product's concept may have been too vague, we could have made more inquiries about the nature of the product during the interviews, thus reaching a better understanding of the challenges regarding productization in the project.

The nature of the interviews must be taken into account when assessing validity. Even

when remaining anonymous, interviewees may omit relevant details if they fear that revealing them endangers their professional relationships or position in the company. Relevant details may also be omitted simply because they are not remembered, or never specifically asked about.

4.5.1 External validity

Single-case studies inspect some specific situation, and they are not meant to be faithfully replicated [GHF00]. Regardless, their generalizability can still be evaluated.

The researcher can improve the generalizability of the case study by selecting a case that is considered typical, instead of selecting the case based on convenience [GHF00]. While practicality was a major factor in our case selection, the case still reflects the typical challenges internal startups face, based on literature. However, the case project cannot be considered a typical project in terms of testing, as testing the product under development was considered especially challenging by the interviewees. We also acknowledge that the product under development was likely more complex to use compared to a typical product developed by an internal startup.

An atypical case can also support generalizations [GHF00]. In our case, the core development team consisted of experienced testing professionals, which could make the case ideal for evaluating the best possible testing effort in an internal startup. However, other factors in our case did not support strong testing effort. Real life cases are likely to be complex and in some ways atypical, and the studied case should be described in depth to provide the best possible understanding of the factors in the case. This is something we have tried to achieve.

Generalizations are best supported by multiple case studies which deal with the same subject [GRW08]. To make valid generalizations, our case study should be compared with other case studies concerning internal startups.

5 Results

In this chapter we present a thematic model based on the interview material, and explore the model's themes and subthemes individually, while highlighting some connections between themes.

Four themes were composed from the interview material, these themes being 1) product development, 2) the parent company, 3) management in general and 4) testing. These themes were further divided into ten subthemes. We proceeded to create a model of the themes and subthemes where the connections between them are mapped (Figure 5).

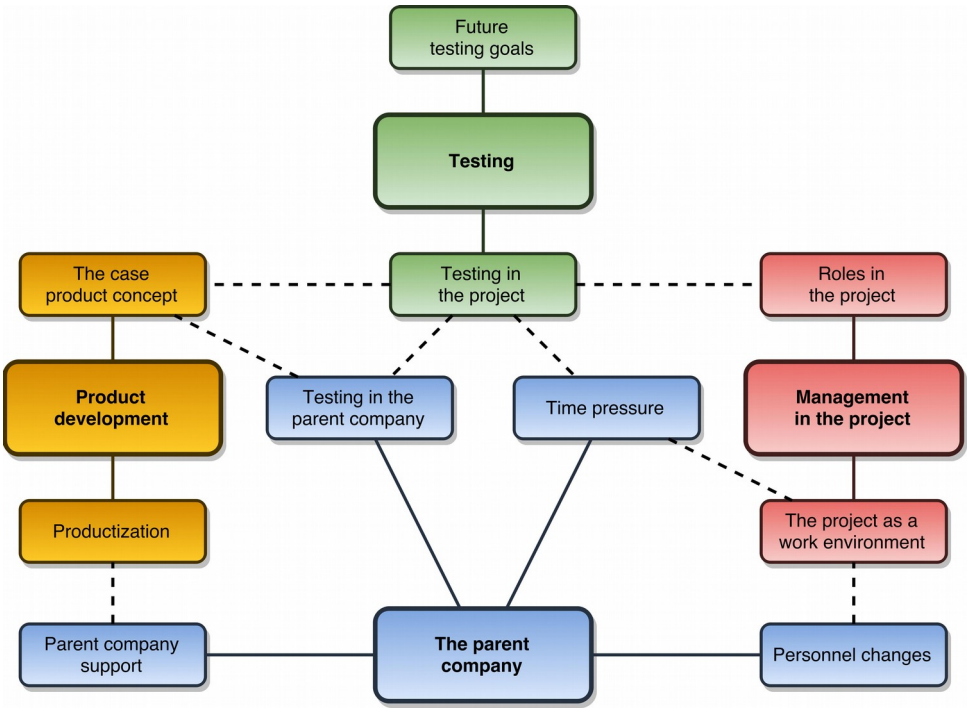

Figure 5: Themes.

In our model the themes are represented by large rectangles while the subthemes connected to them are being represented by smaller rectangles. In the colored version, each theme has their own color which their subthemes share. The subthemes are connected to their themes with a solid line. Connections between subthemes not connected to the same theme are represented by a dashed line.

5.1 Product development

The theme of product development emerged primarily from the interview questions concerning the background of the case. An interviewee pointed out that the company had no standard solution for test automation, continuous integration or continuous delivery either for internal projects or customer projects. Most of the parent company's new projects were required to feature test automation, and the company had no resources to coach all testers who were not well-versed with test automation. One purpose of the case product was to answer to these needs.

The motivation behind the case product is represented in our model by a connection between the case product concept and testing in the parent company.

5.1.1 The case product concept

The concept of the case product was a novel, easy-to-use test automation and continuous integration tool for testers and developers. The case product was conceived as a "self-service" product which could be used with the support of a user guide. Convenience of use of the case product was prioritized over product quality or price. The user interface of the product was supposed to be minimalist and easy to use. The case product was considered to be innovative by the interviewees, and one key team member was unsure how the users would eventually use the product. The planned user experience of the product was based on a use case conceived by the case project team.

The case product was originally supposed to automatically build test environments based on the web address of the customer project repository; an interviewee compared this process to how the Docker software[1] builds virtualization images. The user could configure different features of the case product in a configuration file that the user included in her own project repository.

The case product was designed to be a comprehensive, integrated framework or testing tool with an automated release "pipeline" to facilitate continuous integration with fully automated testing. During the parent company's year-long innovation program the team participated in, more features were added to the case product, including service virtualization and remote computer support. During the whole development process the introduction rate of new features had been rapid, which led to frequent refactoring

1 https://www.docker.com/, retrieved 14 October 2017.

of code. However, the core features of the case product had remained constant.

5.1.2 Productization

The strategy of the parent company was to sell services rather than products, and multiple interviewees noted how hard it is to develop a product in the company. A key team member argued the parent company has no expertise in productization, ie. the development of fully marketable products. Productization, however, was crucially important in the development of the case product, according to the interviewee.

There was an effort to sell the case product to customers as a "beta version" that could be developed in collaboration with the customer. The testers of customer companies were specifically considered as a target group, as they were potential end users. The sales and testing personnel of the parent company were aware of the product, and the product was showcased to some customers. However, the product was considered difficult to sell, especially to customers who already used an established testing process.

Many interviewees considered quality an important aspect of a testing tool such as the case product, especially when the product is offered as a testing solution to a customer. One interviewee stressed that quality should be emphasized in an internal startup just as much as in other projects. A key team member, who later mentioned Lean startup as an inspiration in the project, argued that if the product is developed in collaboration with the customer, the developers must react fast to how the customer uses it. The importance of customer-set priorities in general was emphasized by the interviewees, even though they were not talking about the case product specifically.

5.2 The parent company

The support of the parent company was one theme of the interview questions, and the role of the parent company was a prominent subject in general.

5.2.1 Testing in the parent company

The parent company's testing practices were criticized by several interviewees: when projects ran out of money or time, testing and especially test automation were the first activities to be scaled back or abandoned. Only a small minority of testers in the company used test automation. A former team member pointed out how the projects of

the company, in part because of the customers' wishes, are not agile in practice. Another interviewee remarked that no matter how agile the project is supposed to be, development is often finished before testing even begins.

The potential effect of the parent company's testing practices on the testing in the project are represented in our model by a connection between them.

5.2.2 Parent company support

At least three team members were coached by the parent company's innovation program. The team members were coached to become good salesmen who could efficiently market their idea to the parent company leadership. Most of their time in the program was spent on designing and perfecting slidesets.

The two key team members and a former team member emphasized the autonomy of the project, and a key team member stated that "we do what we want". The team used the parent company's resources in many different ways. The case product sales material was edited and stylized by another unit, and the sales organization of the parent company negotiated the use of the case product with customers. One unit of the parent company provided programmers for the team, and the test organization of the company helped with testing.

The parent company's cloud service was utilized free of charge. However, many interviewees played down the support the project had from the parent company, even referring to it as non-existent. The team could not easily obtain the licenses they needed. One essential tool was used as a free-to-use trial version. One external component used in the case product required a license, but the team had the owner company's permission to use it free of charge.

The parent company's support of the project in the productization effort is represented in our model by the connection between them.

5.2.3 Time pressure

Time pressure had an effect on the project, as the team was under pressure to develop "something you can show". Features were prioritized, and little time was left for testing; this was described as a typical pattern in software development by some interviewees. The maintenance of the existing tests was not performed regularly, and

after a break the tests were often "completely broken".

The interviewees argued more time was needed for testing: a former team member pointed out that when some features are goals for a two-week sprint, the completion of those features does not necessarily leave room for testing. As a key member stated: "In a startup you are under terrible pressure to make something to show and under pressure to get the first customers. Had we done this by the book, I don't know if we still even had anything to show." According to him, it is important to create a Minimum Viable Product fast with one basic, marketable functionality, and he still considered the case product to be at the MVP stage.

The effect of time pressure on the testing in the project is represented in our model by a connection between them.

5.2.4 Personnel changes

The team members did not work exclusively in the project, but had to work in customer projects as well. The customer projects were always prioritized over internal projects in the parent company as they were a direct source of income for the company. A key team member stated that he worked in the project "in addition to other duties", and another interviewee thought the project should have been an exclusive project with a limited time frame. However, the team members could often regulate the amount of time they worked for the customer projects and often chose to work in them because of their own interest.

The parent company provided personnel for the team by moving employees from other units: the interviewees did not consider their "home" subunits important, and instead emphasized the importance of project teams. The criterion of choice for selected personnel was not relevant competence, but availability. The parent company wanted to keep its employees working actively, even in an internal project like the case project. A key team member criticized the selection criterion for the new team members, as he wanted to emphasize relevant competence in the selection.

All but the two key team members were eventually taken out of the project and replaced by new team members. Multiple interviewees suggested this loss of team members was a consequence of the parent company's prioritization of customer projects, and an interviewee stated he was taken out "because I did not bring in money

in that project". All interviewees considered the turnover rate of team members too fast. The key team members were frustrated, as new team members were completely unfamiliar with the project and had to be trained by the key team members. A key team member stated: "if they [the developers] change every month...it's like we don't have any developers", adding that the existing team members had to prioritize all customer projects over the case project.

A former team member stated that a key team member had to fight the parent company to keep developers in the project. According to him, the key team member saved the project from cancellation only by arranging a move to another unit inside the parent company. An interviewee who was not part of the team suggested that it must be a struggle to secure support and resources for the project, as "internal projects probably never get the support they need".

5.3 Management in the project

The management and personnel theme concerns project management in the case project and the professional background and motivation of the personnel working in the project. The theme emerged primarily from the interview questions concerning the interviewees' background and management in the case project.

5.3.1 Roles in the project

When the interviews were conducted, the project had lasted for approximately ten months. While the two key team members had worked in the project continuously, if not exclusively, the other team members had only worked in the project for a few months. Three team members who we interviewed were test automation experts, while one team member was a developer with no background in testing. The team had also experience with customer interface. All interviewees had more than ten years of work experience in the IT industry.

In the case project the team members had flexible roles, but specialized in different tasks. The two key team members were in charge of the project, and one of them was described as an unofficial project manager. The other key team member specialized in developing the technical testing capabilities of the case product. The two key team members were primarily responsible for designing the features of the case product, as well as sales activities and the training of new team members.

The implementation of the features was a shared responsibility of the team, but the non-permanent team members concentrated on either testing or development. All team members were required to actively learn new technology to be able to contribute effectively. A key team member stated that "...you can't say that this is not your responsibility, that you can't do it — you have to learn to do it."

Multiple interviewees argued that in a small startup the roles need to be flexible and all team members have to contribute to testing. A key team member argued that while dedicated testers should be tasked with creating test cases, it is the whole team's responsibility to run them and to monitor quality. The interviewee also argued that very small teams of under ten members should not have dedicated testers or managers; however, a startup team still needs members with special expertise.

A key team member argued that the development team members should be trusted "to be professionals" and to properly test the software under development and verify its quality. A former tester in the team stated that developers can help the tester to design the test cases, but expected testers to be responsible for writing them. Additionally, a former team member remarked how rare good testers are.

A key team member argued every software project should have a "quality lead", who monitors quality in the project and ensures testing is conducted with proper methods and tools. However, because of the commitment to other projects in the parent company, the key team member did not have time to lead testing. A former team member stated that even if all team members contribute to testing, one team member has to establish the test environment, and another interviewee recommended a specialized tester role for the startup team.

The link between the roles employed in the project and testing in the project is represented in our model by a connection between them.

5.3.2 The project as a work environment

While the two key team members were in charge of the project, the other team members agreed that the decision-making in the team was democratic. Communication was open, and ideas were expressed and discussed freely. Team members were trusted, they worked very independently, and there was no strict control or supervision regarding their work. A former team member noted that while he enjoyed working in

the project, he considered it a turbulent environment, referring to the lack of planning. The interviewee stated that while the project had Scrum-style daily meetings and two-week sprints were attempted, the team did not employ agile methodology consistently.

A key team member acknowledged that the project was not very formal, believing more formality, like defined delivery goals, would have made the team more effective. However, in the launch phase the features of the case product were the priority, and setting up formal procedures felt like a burden for the key team members. A former team member argued that startups should have few formal processes in the launch phase, but they are needed when more people join the project. According to the interviewee, formality becomes especially important when the startup gets its first customers.

A former team member pointed out how the "ownership" of the project made the startup project different from customer projects. The team members cared about the case product, and there was a desire to create great software within the team. Even the quality of the software would eventually have been improved, the interviewee presumed.

The effect of time pressure and personnel changes on the project's work environment is represented in our model by a connection between them.

5.4 Testing

Because of the subject of our study, testing was a prominent theme in the interview questions. The theme emerged primarily from the interview questions concerning testing in general and in the case project specifically.

5.4.1 Testing in the project

The interviewees associated testing of the product with multiple potential problems. As the case product was a testing tool where the user interface was not a priority, it was demanding to test — especially as it used new technology like Docker. When the product is refactored over time and new features are added, the existing tests should be updated constantly, which slows the project down.

In contrast to customer projects in the parent company, the requirements of the case project were not as explicitly defined. A former team member stated that the team only

discussed informally the next steps to be taken. Small-scale smoke testing was conducted to verify that the case product was working as intended, but there were no comprehensive testing goals or plans for the product.

Test automation was prioritized highly by several interviewees, and it was considered an integral part of continuous development as well as agile software development by a key team member. An interviewee noted that if test automation is not integrated in the project from the beginning, it is hard to sell to a customer.

Some automated test cases were used to test the case product, but most team members we interviewed considered testing in the project small-scale or even inadequate. Few tests were written and they were seldom run, and there was no agreed "definition of done" for any features. However, the key team members had recently hired a new tester for the project and had a plan to improve test automation in the project.

Code coverage was considered a significant metric in regression testing, although a former team member considered it easy to manipulate. Some unit tests were written for the case product, but the goal of 80 percent coverage had not been reached. Test Driven Development performed with automated test cases was considered an effective way to monitor test coverage by a key team member. However, a former team member considered the effort required for high code coverage inordinate in respect of coverage's value.

Multiple interviewees advocated prioritizing features in the startup's launch phase and improving testing when the project has become more stable, but is not yet ending. However, a former tester in the team argued that testing in the launch phase often inspires new features; he considered testing always to have some value, even when early tests become outdated. Another interviewee argued that the results of every development phase should be verified in some way.

The challenging nature of the product is represented in our model by the connection between testing in the project and the case product concept.

5.4.2 Future testing goals

Interviewees mentioned multiple ambitious goals that the team had not yet implemented. Testing the case product with itself had been an important goal from the beginning of the project, but this was considered a challenging task. A continuous

delivery pipeline was considered important for the development of the case product, especially as the product itself was a testing tool.

When the interviews were conducted, the team had set the continuous delivery pipeline as their goal, and some integration tests had already been written for the product. Test automation was considered an integral part of the pipeline and continuous delivery in general. Testing databases in addition to user interfaces was singled out as an important step by a key team member. A former tester noted that the continuous delivery server used was a poor choice for the product.

6 Discussion

In this chapter, we discuss the results of our study in regard to literature and our research questions.

6.1 RQ1

Our first research question was:

RQ1: How is testing organized in internal startup projects?

Internal startups operate under a constant threat of cancellation. In the case project, there was pressure to present working (or at least visible) features to the parent company representatives, and quality was not emphasized. While the startup team had autonomy over the case product, they did not have sufficient resources to develop and test it. Even the key team members were not allowed to work exclusively in the project, and the other team members, while competent, were typically selected not because of they had expertise required in the startup, but simply because they were available. While the key team members acknowledged that they could have concentrated more on testing, organizing testing is difficult if the project's testing resources can be taken away at short notice.

We suggest that a secure position in the company, likely enabled by a champion inside the company, allows an internal startup to organize its resources, including testing resources. The results of our case study suggests testing measures cannot be demonstrated to parent company representatives as easily as features and slidesets, leading the startup team to prioritize the completion of features over testing. We suggest an internal startup is likely to invest in testing only when the startup members feel they have the parent company's support.

While a secure position enables the startup to organize its testing, we suggest there is a distinct risk that testing is disregarded in internal startups even under favorable circumstances. Test automation specifically is often disregarded in startups, as it is seen as inaccessible and time-consuming. In the case project, test automation and continuous integration was a key aspect of the product, but the team members did not think there was enough time to create test automation to test the product. The agile methodology, including Lean Startup, de-emphasizes specialized roles, suggesting all

team members should be responsible for quality assurance. However, committing to agile practices in principle does not by itself guarantee that team members do actually commit to any quality assurance in practice, especially when fast progress is emphasized in the project. The key members in the project in our case study were experienced testing professionals who actively advanced quality assurance in the company, and the product they were developing was a test automation tool. Even with these foundations, testing of the product itself was overlooked, which we consider a typical development in internal startups. However, the key team members had plans to increase automated testing in the project, which suggests quality can become a more relevant concern in an internal startup as the project matures.

Agile software development de-emphasizes formal processes, as does the startup culture. However, Lean Startup proposes a Build-Measure-Learn -loop to guide xz development, and agile software development in general favors iterative development. Iterative techniques may not be as formal as traditional process models such as the V-model, but they still provide a guideline for development and testing. The essential Measure-part of the Build-Measure-Learn -loop is practically a testing procedure. While the generally informal ethos of agile software development and Lean Startup may inspire an internal startup to forgo formal procedures and roles in testing, disregard of quality assurance is not encouraged by these approaches.

Testing practices in the parent company may culturally influence an internal startup associated with it. However, in the case project, the team members criticized the testing practices of the parent company as inadequate, and the key team members were activists for change in this regard.

6.2 RQ2

Our second research question was:

RQ2: How should testing be organized in internal startup projects?

The question is complex to answer, as the organization of testing should fit the requirements of a specific project. Answering the question requires a specific context where chosen aspects of software development are prioritized over others. However, based on literature and our results, we can make some general suggestions.

From a company's point of view, the motivation behind internal startups is to create a

space where the company's conventions are challenged to foster innovation. To ensure this, the parent company should commit to the internal startup and provide it with resources necessary to test its product in the long term. To effectively organize its testing resources, the internal startup must have resources in the first place.

Based loosely on [CrG09] and the interviews, we suggest that the semi-formal position of a tester or a "quality lead" helps to ensure the project's shared commitment to quality assurance. In a small startup project strict roles are not practical, but having at least one designated team member to emphasize and maintain testing along her other duties can help to ensure that testing is not disregarded.

Traditionally oriented software testing literature emphasizes how testers outside the development team can provide a different point of view compared to testers working directly with the developers. While users with access to the product may help in supplying this outsider perspective, professional testers outside the project can provide valuable insights.

6.3 RQ3

Our third research question was:

RQ3: Is the testing of internal startup projects different from the practices companies usually employ and if it is, why?

Literature suggests testing has a tendency to be overlooked and undervalued in software projects, especially when the stakeholders are acting under time pressure. Based on literature and the results of our case study, we suggest this tendency is even stronger in internal startups because of their lack of established testing procedures and roles as well as their unstable status in the parent company.

In the case of the parent company of the case project, the company's testing practices were criticized as being inadequate by several interviewees. Testing in general and test automation specifically were seen as insufficiently resourced or even neglected in the company projects. However, team members linked the disregard of testing in the case project more to the generally unstable status of the project rather than parent company influence.

7 Conclusions

While Lean Startup grants internal startups a degree of independence, they are dependent on their parent company, and operate under a constant threat of cancellation. The parent company functions as an additional customer, whose approval the internal startup must secure to remain functional. The internal startup is under pressure to demonstrate their worth to the parent company, and testing the startup's product diligently may not be a fast method to achieve this goal. The internal startup also usually has minimal resources at their disposal, which often has an adverse effect on testing. Organizing the startup's human resources effectively is difficult when those resources are provided on a temporary basis, as in the case project.

Internal startups are influenced by agile methodology, where software is typically developed in democratic teams. The team members have no specialized roles, and they are expected to adhere to the agile ideal and take collective responsibility of the product quality. Development is done in short-term iterations which result in a usable product. Testing is not executed in consecutive phases after development, but made an integral and continuous part of the whole development process. This development is only accelerated by the influential DevOps approach.

Internal startups favor light, agile structures such as democratic teams without specialized roles. The agile approach stresses the development team's collective commitment to product quality and testing: this demand is problematic in internal startups, which operate under pressure and with minimal resources. In practice, internal startups tend to answer to the immediate needs of their parent company. Testing, nominally integral to the development process, is overlooked in favor of new features which are used as a proof of the startup's vitality. This relationship is similar to the one between individual startup companies and their investors.

Companies launch internal startups to innovate, but to do this startups require stability, protection, resources, and relative independence. We recommend that a company launching an internal startup takes care in providing the startup the resources necessary for it to develop, test and market their product properly. While there will always be negotiation about the limits of the startup's independence, the reasonable demands of the startup team must be met for the project to be viable. To ensure the internal startup's shared commitment to quality, we also recommend that a

designated tester is included in the startup team.

References

[AJS04] Ahonen, J. J., Junttila, T., & Sakkinen, M. (2004). Impacts of the organizational model on testing: Three industrial cases. Empirical Software Engineering, 9(4), 275-296.

[ABC13] Anand, S., Burke, E. K., Chen, T. Y., Clark, J., Cohen, M. B., Grieskamp, W., ... & Mcminn, P. (2013). An orchestrated survey of methodologies for automated software test case generation. Journal of Systems and Software, 86(8), 1978-2001.

[BLB13] Björk, J., Ljungblad, J., & Bosch, J. (2013, June). Lean Product Development in Early Stage Startups. In IW-LCSP@ ICSOB (pp. 19-32).

[CBG04] Cohen, C. F., Birkin, S. J., Garfield, M. J., & Webb, H. W. (2004). Managing conflict in software testing. Communications of the ACM, 47(1), 76-81.

[CrJ02] Craig, R. D., & Jaskiel, S. P. (2002). Systematic software testing. Artech House.

[CrG09] Crispin, L., & Gregory, J. (2009). Agile testing: A practical guide for testers and agile teams. Addison-Wesley Professional.

[CrA10] Crossan, M. M., & Apaydin, M. (2010). A multi-dimensional framework of organizational innovation: A systematic review of the literature. Journal of management studies, 47(6), 1154-1191.

[CrD11] Cruzes, D. S., & Dyba, T. (2011, September). Recommended steps for thematic synthesis in software engineering. In Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on (pp. 275-284). IEEE.

[DeS13] Deak, A., & Stålhane, T. (2013, March). Organization of testing activities in Norwegian Software Companies. In Software Testing, Verification and Validation Workshops (ICSTW), 2013 IEEE Sixth International Conference on (pp. 102-107). IEEE.

[DNB12] Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. The

Journal of Systems & Software, 6(85), 1213-1221.

[Dru14] Drury-Grogan, M. L. (2014). Performance on agile teams: Relating iteration objectives and critical decisions to project management success factors. Information and Software Technology, 56(5), 506-515.

[EGH16] Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016). DevOps. IEEE Software, 33(3), 94-100.

[EBT13] Edison, H., Bin Ali, N., & Torkar, R. (2013). Towards innovation measurement in the software industry. Journal of Systems and Software, 86(5), 1390-1407.

[ElS16] Elbanna, A., & Sarker, S. (2016). The Risks of Agile Software Development: Learning from Adopters. IEEE Software, 33(5), 72-79.

[EWA15] Edison, H., Wang, X., & Abrahamsson, P. (2015, May). Lean startup: why large software companies should care. In Scientific Workshop Proceedings of the XP2015 (p. 2). ACM.

[Ger04] Gerring, J. (2004). What is a case study and what is it good for?. American political science review, 98(2), 341-354.

[GHF00] Gomm, R., Hammersley, M., & Foster, P. (Eds.). (2000). Case study method: Key issues, key texts. Sage.

[GRW08] Gibbert, M., Ruigrok, W., & Wicki, B. (2008). What passes as a rigorous case study?. Strategic management journal, 29(13), 1465-1474.

[GZN10] Guo, P. J., Zimmermann, T., Nagappan, N., & Murphy, B. (2010, May). Characterizing and predicting which bugs get fixed: an empirical study of Microsoft Windows. In Software Engineering, 2010 ACM/IEEE 32nd International Conference on (Vol. 1, pp. 495-504). IEEE.

[HyK14] Hynninen, P., & Kauppinen, M. (2014, August). A/B testing: A promising tool for customer value evaluation. In Requirements Engineering and Testing (RET), 2014 IEEE 1st International Workshop on (pp. 16-17). IEEE.

[KRT09] Karhu, K., Repo, T., Taipale, O., & Smolander, K. (2009, April). Empirical observations on software testing automation. In Software Testing Verification and Validation, 2009. ICST'09. International Conference on (pp. 201-

209). IEEE.

[KTS09] Kasurinen, J., Taipale, O., & Smolander, K. (2009, December). Analysis of problems in testing practices. In Software Engineering Conference, 2009. APSEC'09. Asia-Pacific (pp. 309-315). IEEE.

[KoP99] Koomen, T., & Pol, M. (1999). Test process improvement: a practical step-by-step guide to structured testing. Addison-Wesley Longman Publishing Co., Inc..

[Lea14] Leavy, P. (Ed.). (2014). The Oxford handbook of qualitative research. Oxford Library of Psychology.

[LiM16] Lindgren, E., & Münch, J. (2016). Raising the odds of success: the current state of experimentation in product development. Information and Software Technology, 77, 80-91.

[MRR07] Martin, D., Rooksby, J., Rouncefield, M., & Sommerville, I. (2007, May). 'Good' organisational reasons for 'Bad' software testing: An ethnographic study of testing in a small software company. In Proceedings of the 29th international conference on Software Engineering (pp. 602-611). IEEE Computer Society.

[MSB16] Mårtensson, T., Ståhl, D., & Bosch, J. (2016). Continuous Integration Applied to Software-Intensive Embedded Systems–Problems and Experiences. In Product-Focused Software Process Improvement: 17th International Conference, PROFES 2016, Trondheim, Norway, November 22-24, 2016, Proceedings 17 (pp. 448-457). Springer International Publishing.

[MII12] Mäntylä, M. V., Itkonen, J., & Iivonen, J. (2012). Who tested my software? Testing as an organizationally cross-cutting activity. Software Quality Journal, 20(1), 145-172.

[NeS13] Neely, S., & Stolt, S. (2013, August). Continuous delivery? Easy! Just change everything (well, maybe it is not that easy). In Agile Conference (AGILE), 2013 (pp. 121-128). IEEE.

[PGU14] Paternoster, N., Giardino, C., Unterkalmsteiner, M., Gorschek, T., & Abrahamsson, P. (2014). Software development in startup companies: A systematic mapping study. Information and Software Technology, 56(10),

1200-1218.

[Rie11] Ries, E. (2011). The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses. Crown Business.

[SAN17] Silva, A., Araújo, T., Nunes, J., Perkusich, M., Dilorenzo, E., Almeida, H., & Perkusich, A. (2017). A systematic review on the use of Definition of Done on agile so ware development projects.

[SLS11] Spillner, A., Linz, T., & Schaefer, H. (2011). Software Testing Foundations: A Study Guide for the Certified Tester Exam.

[Sut00] Sutton, S. M. (2000). The role of process in software start-up. IEEE Software, 17(4), 33-39.

[TaM14] Tamburrelli, G., & Margara, A. (2014, August). Towards Automated A/B Testing. In International Symposium on Search Based Software Engineering (pp. 184-198). Springer International Publishing.

[VST09] Verner, J. M., Sampson, J., Tosic, V., Bakar, N. A., & Kitchenham, B. A. (2009, April). Guidelines for industrially-based multiple case studies in software engineering. In Research Challenges in Information Science, 2009. RCIS 2009. Third International Conference on (pp. 313-324). IEEE.

[WES12] Wiklund, K., Eldh, S., Sundmark, D., & Lundqvist, K. (2012, April). Technical debt in test automation. In Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on (pp. 887-892). IEEE.

[WWW17] http://glossary.istqb.org, retrieved 6 August 2017

Appendix 1. The interview question set

Interviewee background

Interview Question 1. What is your name?

IQ2. What is the position you usually work in?

IQ3. How much work experience you have from this field, in years?

IQ4. What kind of unit and team you usually work in?

Case project background

IQ5. Describe the case project in general, please.

IQ6. What was your role in the project, and how was it different from what you usually do?

IQ7. Was there a project manager in the case project, official or unofficial?

IQ8. Were you directed in any way during the project?

Parent company support

IQ9. Did the project team had any assistance from other units or organizations during the case project?

IQ10. Did the parent company limit the time you were allowed to use for the project?

IQ11. Do you feel the project had the parent company's support? For example, how was the project prioritized by the parent company, did you receive i.e. the hardware and product licenses you required?

Testing in the case project

IQ12. How was the testing organized in the case project in a general sense? What kind of goals were there for testing?

IQ13. Were the testing goals met? If not, what actions would have been required to meet them?

IQ14. Was any test automation used in the project? Did you use it yourself in any way?

IQ15. How were the project's testing practices different compared with the parent

company's standard testing practices? Think about your last, non-startup project in the company, for instance. (If there are differences) Can you think of any reasons for the differences?

Best practices in testing

IQ16. How should testing be organized in an internal startup such as the case project?

IQ17. How important do you think wide test coverage and test automation are in an internal startup project?

Closing remarks

IQ18. Do you have anything to add to your answers? Do you have something else you want to say, related to the subject matter of our study?