The Journal of Systems and Software 167 (2020) 110599

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Achieving agility and quality in product development - an empirical study of hardware startups

Vebjørn Berg[a,*], Jørgen Birkeland[a], Anh Nguyen-Duc[b], Ilias O. Pappas[a,c], Letizia Jaccheri[a]

[a] Department of Computer Science, Norwegian University of Science and Technology, Sem Sælands vei 9, Trondheim 7034, Norway
[b] Department of Business and IT, University of South-Eastern Norway, Lærerskoleveien 40, Notodden 3679, Norway
[c] Department of Information Systems, University of Agder, Universitetsveien 25, Kristiansand 4630, Norway

ARTICLE INFO

Article history:
Received 6 January 2019
Revised 16 March 2020
Accepted 8 April 2020
Available online 22 April 2020

Keywords:
Startup
Hardware startup
Software engineering
Product development
Empirical research

ABSTRACT

Context: Startups aim at scaling their business, often by developing innovative products with limited human and financial resources. The development of software products in the startup context is known as opportunistic, agility-driven, and with high tolerance for technical debt. The special context of hardware startups calls for a better understanding of state-of-the-practice of hardware startups' activities. Objective: This study aimed to identify whether and how startups can achieve product quality while maintaining focus on agility. Method: We conducted an exploratory study with 13 hardware startups, collecting data through semi-structured interviews and analysis of documentation. We proposed an integrative model of agility and quality in hardware startups. Results: Agility in hardware startups is complex and not achieved through adoption of fast-paced development practices alone. Hardware startups follow a quality-driven approach for development of core components, where frequent user testing is a measure for early debt management. Hardware startups often lack mindset and strategies for achieving long-term quality in early stages. Conclusions: Hardware startups need attention to hardware quality to allow for evolutionary prototyping and speed. Future research should focus on defining quality-driven practices that contribute to agility, and strategies and mindsets to support long-term quality in the hardware startup context.

1. Introduction

Startups, newly created companies producing cutting-edge technology, are an important source of technology innovation, and have a significant impact on the wave of digital transformation (Jacobson et al., 2017). Despite stories of successful startups, most of them fail, primarily due to self-destruction rather than competition (Crowne, 2002; Marmer et al., 2011). Without previous operational experience, startups often need to learn how to establish new roles, new connections to external stakeholders, and new processes and practices (Stinchcombe, 2000; Abatecola et al., 2012). In a startup company developing high-tech products, besides personal trait of startup founders and financial and market factors (Giardino et al., 2015; Aldrich and Auster, 1986; Van Gelderen et al., 2005), product development is also a key factor characterizing the development of the startup (Unterkalmsteiner et al., 2016; Giardino et al., 2015; Tripathi et al., 2016; Giardino et al., 2016; 2014a). For instance, software research has shown interest in achieving effective Minimum Viable Products (Nguyen-Duc and Abrahamsson, 2016) and managing technical debt (Giardino et al., 2016) in the startup context. Even though the obstacles to success gradually become known and aware to entrepreneurs, the startup context poses several unique challenges to traditional product development and innovation methods (Unterkalmsteiner et al., 2016; Nguyen-Duc et al., 2016).

The part of startup ecosystems that is relatively little explored in research is hardware startups. They include startup companies developing products and services with a value proposition based on an integral solution of software and hardware components (DiResta et al., 2015; Jacobson et al., 2017). Hardware is a physical, tangible part of a system, or a system of systems (e.g., sensors, gateways, connectivity components, wearable devices, mobile phones), while software is a code-based, intangible part of the system (e.g., operating systems, server-side scripts, databases, algorithms). A typical example for a modern hardware system is a connected house, where the hardware part is implemented to measure, collect and transmit data, and the software part is

* Corresponding author.
E-mail addresses: vebjornb@netcompany.com (V. Berg), jorgen.birkeland1@gmail.com (J. Birkeland), anh.nguyen.duc@usn.no (A. Nguyen-Duc), ilias.pappas@uia.no, ilpappas@ntnu.no (I.O. Pappas), letizia.jaccheri@ntnu.no (L. Jaccheri).

https://doi.org/10.1016/j.jss.2020.110599

used to coordinate the operations of hardware, store and process the collected data. The barriers for starting a hardware company have never been lower, a result of the advanced development of hardware technology. Rapid prototyping, decreased component costs, small-batch manufacturing, and fundraising platforms have renewed the interest for hardware startups (DiResta et al., 2015; Wei, 2017).

Hardware startups add additional complexity to software startups as they need to handle the development and integration of hardware parts into the offered products (Nguyen-Duc et al., 2018). Hardware products usually need to be secured and safe, which puts a focus on ensuring quality attributes of delivered products. Moreover, the quality of the whole product relies on the quality of its integrated components, both software and hardware modules. While it is known that software startups focus on speed and agility, remaining low priority on quality assurance, it is not known if the same practices occur in hardware-related product development. While knowledge from development of embedded products in established companies can be relevant (Kaisti et al., 2013; Albuquerque et al., 2012), the "newness and smallness" nature of startups calls for an investigation and further, an adaption of existing methodologies and practices that are suitable to startup context (Bosch, 2016).

Software startups are known for fast-paced development, with ability to handle uncertainty, react to changes in product and business development, and introduce flexibility in the process (Garbajosa et al., 2017). The concept of agility in hardware startups might be different from pure software development, as hardware development typically involves a long development cycle and depends on a larger set of third-party vendors. The relationship between agility and quality might be more critical in some circumstances, for instance startup companies who deliver quality-driven products. For example, the Norwegian startup Prediktor Medical AS develops a glucose smartwatch that measures glucose level without penetrating people's skin. The product was quality-driven and has been developed under a market-pressure with a promised launch time. A recent industry survey also calls for systematic adoption of product development methodologies in hardware startups (Nguyen-Duc et al., 2018).

To this end, we seek to *create a better understanding of work-practices in hardware startups by investigating the role of engineering activities, from idea conceptualization to a launched product. In particular, we will investigate factors influencing agility and quality, and explore commonalities and challenges.* As mentioned by Jacobson et al. (2017), literature regarding methods for hardware product development is scarce. We aim at exploring how agility and quality are managed in practice. This has motivated the following research question:

RQ How do hardware startups achieve both agility and product quality during product development?

This paper presents the results from a qualitative survey investigating 13 early-stage European hardware startups. The work contributes to startup engineering research by focusing on hardware-intensive product development. The research provides early empirical evidence to agility in hardware startups, and simple quality-aware practices in the context of restricted resources. The work also builds the foundation for researchers and practitioners to further explore hardware startup engineering, which is still in a nascent stage.

The remainder of this paper proceeds as follows: Section 2 introduces the background of the study and relevant theoretical frameworks. Section 3 presents the research method undertaken and potential threats to the validity. Section 4 reports the results of the study, including transcribed citations from the participants. Section 5 discusses the results in relation to the research questions. Section 6 concludes the paper by answering the research questions and proposing directions for future work.

2. Background

2.1. The context of high-tech startup companies

The term "startup" has been defined differently across various principles (Steininger, 2019; Sutton, 2000; Ghezzi, 2018; Unterkalmsteiner et al., 2016; Crowne, 2002). From the recurrent themes on startups, high-tech startups share common characteristics of organizations focusing on the creation of software-intensive products, with little or no operating history, aiming to grow by aggressively scaling their business in highly scalable markets (Giardino et al., 2016). The context of startups is long understood as a special organizational state. New companies generally involve new roles, and the "coordination of strangers" scenario often lead to low quality of performance (Stinchcombe, 2000; Abatecola et al., 2012). Sommer et al. (2009) highlighted that new companies often do not correctly foresee real opportunities or the best ways of addressing them, and so are forced to adapt and modify their approach over time. Giardino et al. (2014b) revealed that startups often fail to achieve the problem-solution fit during their execution. From early-stage, startups increase their learning curve and foster the establishment of survival determinants (i.e., successful practices and procedures) (Abatecola et al., 2012; Hodgson and Knudsen, 2004).

2.2. Software product development in startup companies

Startups generally develop products in high-potential target markets without necessarily knowing *what* customers want (Blank, 2013b; Rafiq et al., 2017). Increasingly more industries experience that new technologies become available to all players at the same time, hence the benefits of technology-driven innovations decrease. This has led companies to prioritize customer-driven development, which involves identifying new and unknown customer needs as well as meeting known needs (Bosch, 2016). This relates to market-driven software development, where requirements tend to be (1) invented by the software company, (2) rarely documented (Karlsson et al., 2002), and (3) validated only after the product is released in the market (Carmel, 1994; Dahlstedt, 2003; Keil and Carmel, 1995; Rafiq et al., 2017). Products not meeting customer needs are common, resulting in failure of new product releases (Alves et al., 2006). There exist several entrepreneurial theories and frameworks that can guide practitioners in their pursuit to lasting business growth, including "Effectuation Theory" (Sarasvathy, 2001), "Discovery and Creation" (Alvarez and Barney, 2007), the customer development approach introduced by Blank (2013a), and the Lean Startup (Ries, 2011).

Research on software engineering depicts that startup companies prefer to prioritize time and cost over product quality (Yau and Murphy, 2013), neglecting traditional process activities like formal project management, documentation, and testing (Giardino et al., 2016). Shortcuts taken in product quality, design, or infrastructure can inhibit validated learning (Ries, 2011), in a context where customized development practices are necessary to manage the challenges posed by customer development methods. Inadequate use of software engineering practices might be a significant factor leading to the high failure rates of software startups (Klotins et al., 2015).

Entrepreneurs are in general aware of the significance of how their products are built. Even though studies have found that startups are either reluctant to introducing process (Coleman and O'Connor, 2008), or that they use their own mix

of Agile and ad-hoc methods (Giardino et al., 2014a), many startups emphasized the importance of having good practices in building their products (Sutton, 2000; Giardino et al., 2014a). Small early-stage software startups don't experience the same challenges as larger, more experienced companies, and the cost and time of implementing a rigorous Agile methodology may not provide big enough benefits (Yau and Murphy, 2013).

2.3. Agility in product development

Agility as a concept is multi-facet and in many cases refers to the ability of an organization, a team, or a project to react to changes occurred to them (Conboy, 2009). In a general sense, agility can be defined as "the capability to react and adapt to expected and unexpected changes within a dynamic environment constantly and quickly, and to use those changes (if possible) as an advantage" (Bohmer and Lindemann, 2015). In software development, Agile methods have proven to be a powerful tool when the goal is to build a successful, profitable business model (Cunningham et al., 2001). When a company needs to quickly address market and customer needs, Agile processes have proven to be much more effective than traditional high-ceremony processes (Wasserman, 2016). Since the birth of the Agile Manifesto (2001), with stated principles and practices of Agile methodology (Beck et al., 2001), it has become a popular set of practices in the software industry to replace traditional, rigid, and heavy software development processes.

During the last decades, Agile in software engineering has been an extensive research area with an enormous amount of literature (Dybå and Dingsøyr, 2008; Conboy, 2009; Abrahamsson et al., 2010; Díaz et al., 2011; Misra et al., 2012; Jalali and Wohlin, 2010; Da Silva et al., 2011). Existing studies provide the introduction and adoption of Agile methods and their variance in different organizational settings. They do not agree on a unified view of current practices, but offer a broad picture of experience and some contradictory findings (Dybå and Dingsøyr, 2008). Benefits were reported in the following areas: customer collaboration, work processes for handling defects, learning in pair programming, thinking ahead for management, focusing on current work for engineers, and estimation (Dybå and Dingsøyr, 2008). A recurring theme in studies on Agile development is human factors (e.g., team dynamics, team coordination, customer involvement, etc.) and their influence on Agile development. Much research reports experience of combining Agile development with other Software Engineering paradigms, such as distributed teams (Jalali and Wohlin, 2010), product line development (Misra et al., 2012), and user-centered design (Díaz et al., 2011). The combination of product line development, with the focus on upfront investments, planning, design, and Agile methods, with the highlight of rapid and frequent changes, attention to the design is found challenging (Misra et al., 2012). Several practices are investigated in the fusion of Agile methods into more rigid processes, including release planning (Hanssen and Fægri, 2008) and the bottom-up application driven approach with automated acceptance tests (Ghanam and Maurer, 2010).

Research suggests that Agile methods are suitable for software startups, as iterative development approaches are adaptive, with short lead time (Pantiuchina et al., 2017; Paternoster et al., 2014). The adoption of formal sets of Agile practices and methods in startups is limited, often due to an excessive amount of uncertainty and high time-pressure (Giardino et al., 2014a). Startups often use a tailored version of Agile development, in many cases, the quick combination of Agile and other methodologies.

2.4. Engineering processes for embedded system development

Research on development processes in hardware startups is rare, where exploration of state-of-practice is limited to a few studies (Nguyen-Duc et al., 2018). The processes and practices for developing hardware-relevant products have been reported in literature about embedded system engineering, which concerns about application-specific computing devices consisting of hardware and software components (Ronkainen et al., 2002). Current knowledge on development processes of hardware-related products in established companies is rarely transferred to hardware startups' product development, as the startup context is unique and special (Nguyen-Duc et al., 2018; Ronkainen and Abrahamsson, 2003). In the embedded domain, hardware sets strict requirements to software. Development of hardware-intensive systems require simultaneous development of hardware-components and hardware-related software (Ronkainen et al., 2002). Since software allows for frequent updates and releases, the system architecture often seeks to separate hardware from software to allow for two largely independent release processes (Bosch, 2016). This is illustrated in Fig. 1 where hardware and related software development are distinct processes requiring constant communication and interconnected testing and verification.

Ronkainen et al. (2002) found four main characteristics of hardware and related software development, including (1) hard real-time requirements, (2) experimental work, (3) documentation requirements, and (4) testing.

1. Hard real-time requirements (e.g., data throughput rates, cycle counts, or function call latency) mean that if software doesn't meet requirements, further system operation may be at risk. Hardware simulations can help determine the precise operation of hardware without producing an expensive prototype and even enable testing of the hardware-software co-operation.
2. Hardware-oriented software development is experimental by nature, and developers need to understand the whole system to deal with all uncertainties related to changes in hardware and how software affects the whole system. Every requirement cannot be known and every decision cannot be made before writing software. Developers should utilize an iterative development approach to deal with all ambiguities of hardware-related software development.
3. The communication among hardware and software developers must work to implement the hardware-software interface efficiently. Information has to be explicit and relies heavily on exact documentation to minimize information loss between iterations. However, due to the vast amount of experimental work, too much documentation is not feasible in early stages of product development.
4. Testing is an essential activity both due to reliability and device autonomy requirements, and regression tests to ensure parallel development doesn't drift. In addition to independent software and hardware tests, checking the right interaction between hardware and software (i.e., co-verification) is important to ensure the system works as intended.

Recent advancement in hardware technology suggest that Agile practices also could be used in the embedded domain (Kaisti et al., 2013). Although Agile methods and practices may have a positive impact (e.g., decreased development time and reduced error rates) on product development, the use of Agile in the embedded domain is not widespread (Albuquerque et al., 2012). There is a need for a coherent understanding of how Agile methodologies best fit to embedded systems development in the startup context, and how such practices can reduce costs and efforts in different phases of the development process (i.e., requirement management, design, and architecture).

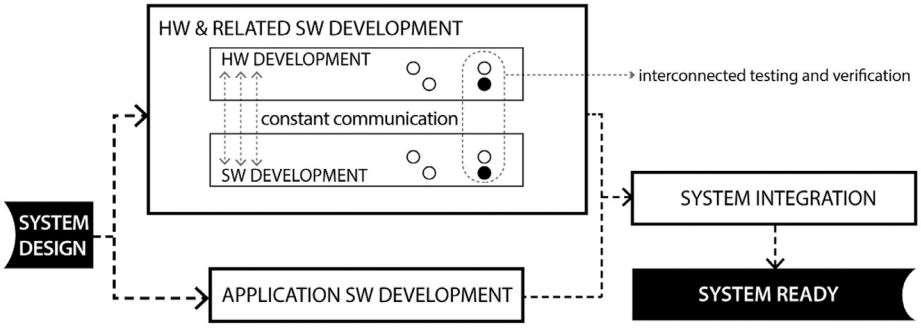


Fig. 1. Hardware-software co-design process (Ronkainen et al., 2002).

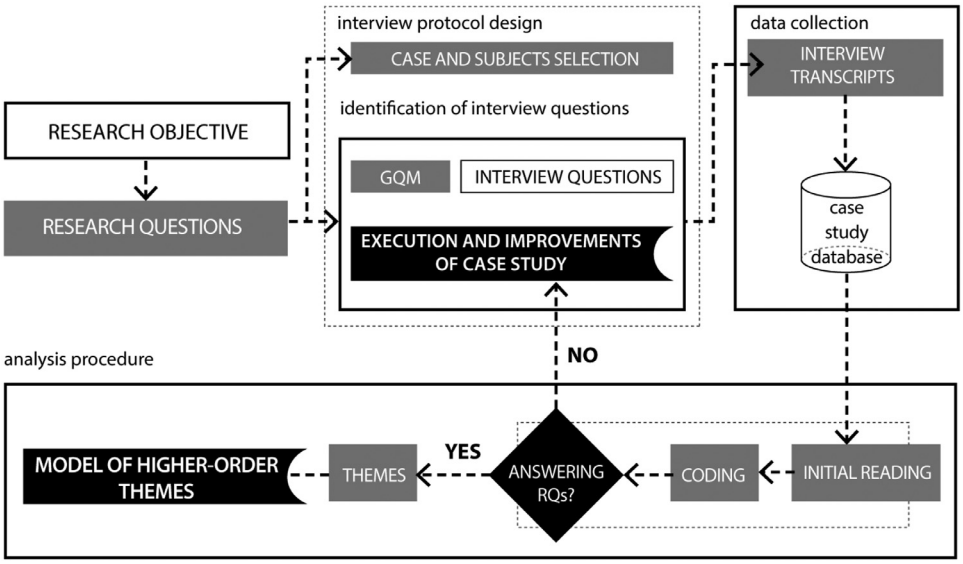


Fig. 2. Research process.

3. Research methodology

Software startup engineering research is to a great extent concerned with investigating the development, operation, and maintenance of software and hardware products in startup companies. In order to gather and to interpret evidence for answering our research questions, we devised a qualitative approach. The goal of qualitative research is to investigate and understand phenomena within their real-life context (Robson, 2002). Dependent on the in-depth knowledge in a case, the qualitative research can have a narrow focus on a few case studies, or a broader scope as a qualitative survey (Robson, 2002; Andersson and Runeson, 2002; Walsham, 1995). As the study's overall goal is to characterize current status of adopting agility and quality-driven practices in a population of hardware startups, a qualitative survey appears to be suitable, especially when there is a limited capacity for capturing insight data from a number of companies in a short time (Andersson and Runeson, 2002). Robson classified four types of research purposes (Robson, 2002):

- Exploratory - understanding what is happening; to seek new insights.
- Descriptive - portraying a situation or phenomenon.
- Explanatory - seeking an explanation of a situation or a problem, mostly but not necessary in the form of a causal relationship.
- Improving - trying to improve a certain aspect of the studied phenomenon.

In line with the non-deterministic nature of product development in the startup context (Nguyen-Duc et al., 2015) (i.e., contexts for product development are highly influenced by team, financial, market situations and entrepreneurial approaches), and with the exploratory nature of our research question, we exploratively investigate multiple startups. Klein and Myers differentiate three types of research perspectives, positivist, critical, and interpretive (Klein and Myers, 1999; Walsham, 1995). Positivist studies search evidence for testing hypotheses, drawing inferences from a sample; critical studies identify social critique, and interpretive studies attempt to understand phenomena through participants' interpretation of their context. In this research, we investigate a phenomenon that integrates human factors and engineering concepts. Hence, we adopted the interpretive view and collected data from semi-structured interviews.

There are several possible levels of analysis (e.g., individual, artifact, team, project and company). We chose project as a suitable level of analysis, as this study concerns about product development activities and processes, with certain expectations about the interactions between the products and their contextual environments. The focus of our interviews is startups' single projects that leads to the launching of their core products. Fig. 2 illustrates all steps of the research process.

Table 1
Startup channels.

Channel	Description	Link
Innovation Center Gløshaugen	The center is located at campus Gløshaugen, and houses various early-stage high-tech startups, mainly to support innovative students.	www.ntnu.no/ig
NTNU Accel and FAKTRY	NTNU Accel is a uni-based accelerator for promising startups. FAKTRY is an incubator which is part of Accel, and houses various hardware startups.	www.ntnuaccel.no, www.faktry.no
Our professional networks	Italian companies (S13), Spanish and Dutch companies (S11)	
OsloTech and StartupLab	OsloTech manage Oslo Science Park, including incubator StartupLab which has supported more than 200 startups since 2012.	www.oslotech.no www.startuplab.no
The Hub	The Hub is a community platform which gives an overview of Norwegian and Nordic startups. Via the platform, startups can get assistance with recruitment and connection with investors.	www.hub.no

3.1. Companies and subjects selection

Our research relies on theoretical sampling: purposive, non-probabilistic samples which are typically small, as a single observation is sufficient for inclusion in the coding system. Researchers identify key participant, for instance, CEO, CTO or key engineers who has access to important information. To select appropriate participants, we chose criteria, as suggested by Runeson and Höst (2009). Startups were relevant for inclusion in the study if they met the following criteria:

- The startup has at least two members, so product development is not an individual activity.
- The startup has been active for at least six months, so their experience can be relevant.
- The startup develops products or services that include both hardware and software parts.
- The startup has a first running prototype so it's engineering practices are relevant.

Our sample in the survey is comprised of 13 hardware companies. They represent a diverse selection of application domains, product types and company characteristics, although they are not systematically sampled from any larger distribution.

People from the relevant startups were eligible for participation if they had experience and/or knowledge about software and/or hardware development. If the candidate met the criteria, he/she was regarded as qualified for contributing to the research study.

Via professional networks of co-authors of this work, we identified several potential sources of contacts, which are co-working spaces, incubator programs, and technology parks. The five different channels used to find relevant startups are (1) Innovation Center Gløshaugen, (2) NTNU Accel and FAKTRY, (3) our co-authors' professional networks, (4) OsloTech and StartupLab, and (5) The Hub. Table 1 provides an overview of the different communication channels and can help other researchers to find and contact startups (Table 2).

There was a mix of startups originated from academica (7 out of 13 companies), entrepreneurs (5 out of 13 companies), and industry spin-off (1 out of 13 companies). The investigated startup founders have varied industrial experience, ranging from 1 to more than 10 years. Regarding entrepreneurial experience, five startups are first time startups. The other eight startups have experienced some failure before. Regarding the background of the interviewees, the majority (12 out of 13 companies) have technical backgrounds that are relevant for developing products (Table 3).

3.2. Data collection procedure

Our chosen data collection method was interviews, identified as an efficient method for answering research questions in explorative studies (Oates, 2005). The semi-structured approach enabled discovery of unforeseen information as interviewees could express themselves more freely, and fitted both the time constraints of the project and the availability of startup companies. We followed a questionnaire guiding the data collection process.

Table 2
Interviewee descriptions.

Company	Role	Background	Gender
Startup 1 (S1)	CEO	Industrial Engineer	M
Startup 2 (S2)	CTO	Informatics	M
Startup 3 (S3)	CTO	Computer Science	M
Startup 4 (S4)	Hardware developer	Cybernetics	M
Startup 5 (S5)	CTO	Electronics	M
Startup 6 (S6)	Software developer	Informatics	M
Startup 7 (S7)	CEO	Electronics	M
Startup 8 (S8)	CEO	Mechanical Engineer	M
Startup 9 (S9)	CEO	Entrepreneurship	M
Startup 10 (S10)	Software developer	Computer Science	M
Startup 11 (S11)	CEO	Computer Science	M
Startup 12 (S12)	CEO	Electronics	M
Startup 13 (S13)	Software developer	Computer Science	M

- Section 1: Warm-up
  1. Tell us about your company at the current stage
  2. What was the original ideas?
- Section 2: Agility and Agile practices
  1. Have you heard about, or used any of the methodologies: Agile, Lean Startup?
  2. How is the methodology implemented?
  3. How do external dependencies influence product development?
  4. How do you balance hardware and software development?
  5. How do you manage documentation?
- Section 3: Quality and Quality assurance
  1. How do you manage product quality?
  2. When did you last refactor the codebase?
  3. To what extent do you reuse components of earlier prototypes?
  4. How do you perform hardware and software testing?
  5. When do you start writing tests?
- Section 4: Closing-up
  1. What would you do differently with the product development?
  2. Any final interesting remarks?

The first and second researcher were in direct contact with the subjects, hence the data collection process can be regarded as a first degree data collection technique. First degree data collection requires a significant effort, but allowed both researchers to control what data was collected, ensuring that all pre-defined interview questions were answered sufficiently and exploring new directions by asking follow-up questions (Runeson and Höst, 2009). Both the first and second author attended all interviews to avoid one single interpretation of the respondent's perspective and in-

Table 3
Startup description.

Company	Product	Current Stage	Founded	Location	# of employees
Startup 1 (S1)	Smart gloves	Startup	2016	Norway	18
Startup 2 (S2)	Medtech biosensor	Startup	2017	Norway	5
Startup 3 (S3)	Physical exercise game	Stabilization	2016	Norway	5
Startup 4 (S4)	Unmanned aircraft system	Startup	2016	Norway	7
Startup 5 (S5)	Advanced noise cancellation	Startup	2017	Norway	5
Startup 6 (S6)	Medtech hydration monitoring	Startup	2016	Norway	10
Startup 7 (S7)	LPG management system	Stabilization	2016	Norway	8
Startup 8 (S8)	Cable cam system	Stabilization	2016	Norway	10
Startup 9 (S9)	Digital piggy bank	Startup	2017	Norway	5
Startup 10 (S10)	Collaborative camera	Growth	2014	Norway	50
Startup 11 (S11)	Interactive children's toy	Startup	2015	Netherlands	8
Startup 12 (S12)	3D-printer control board	Growth	2009	Norway	1
Startup 13 (S13)	Sensors for IoT	Growth	2007	Italy	25

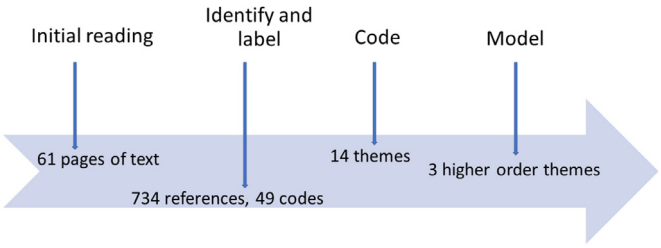


Fig. 3. Thematic Synthesis Process (Cruzes and Dybå, 2011).

sight on topics, as qualitative data often can be rich and broad, but less precise.

Before the interviews, we looked into the participants' business background, either through their company websites or other relevant incubator or accelerator websites. Additionally, most participants answered a simple questionnaire prior to interviews where they filled out basic information about themselves and the company (Appendix B). These measures allowed for more efficient interviews as the first and second author possessed more knowledge about the interviewee and could use less time on formalities. Initial company analysis allowed for a holistic understanding of each company and provided stronger evidence for the conclusions drawn from the interviews. Each interview lasted between 40 min and 1 h. Table 3 presents key facts about the investigated companies. The size of the company provides insight into the required need for development process and managerial overhead. The "Current stage" is adapted from the paper (Crowne, 2002), as applied in the systematic mapping study by Berg et al. (2018), representing the stage of the startups at the time of the interviews. The startup stage refers to the period between product conception and the first sale. The stabilization phase starts when the first customer receives the product, while the growth phase begins when a product is delivered to a new customer without disturbing the development team.

3.3. Analysis procedure

We applied the thematic synthesis process which is a codes-to-theory model for qualitative research (Cruzes and Dybå, 2011). The objective of our thematic synthesis process was to both answer the research questions and come up with a model of higher-order themes describing development strategies in hardware startups, focusing on aspects that are unique from software startups. The main steps of the process are illustrated in Fig. 3.

3.3.1. Initial reading

The first step of the analysis process was to read through the transcribed interviews to generate initial ideas and identify possible patterns in the data. All interviews were transcribed shortly after they were conducted to ensure the actual meaning of interviewees' answers. All authors discussed the interviews, creating a mind map of central concepts relevant to hardware startups.

3.3.2. Coding process

To generate initial codes, the first and second author applied a descriptive coding technique (Saldaña, 2015), to identify interesting concepts, categories, or other findings in a systematic way across the data set. Descriptive coding helped organize and group similar data into categories, which is the first step towards the creation of themes.

The coding process followed an integrated approach (Saldaña, 2015). This allowed us to avoid coding data out of context, while at the same time identifying what the text was saying rather than what we wanted to see. We applied an iterative coding process, to allow for simultaneous data collection and analysis (Runeson and Höst, 2009). The coding process resulted in a total of 49 codes and 734 references from 13 interviews. The first iteration involved coding the data from the four first interviews. A total of 29 codes were generated from 416 references. The codes were examined by the first, second, and third author. Lessons from the evaluation were implemented in the next interviews to generate relevant codes. For the second iteration, we classified text into the codes from the first iteration, while at the same time generating new codes in an inductive manner.

3.3.3. Codes into themes

A theme can be seen as a way of grouping initial codes into a smaller number of sets, to create a meaningful whole of unstructured codes (Cruzes and Dybå, 2011). We divided related codes into categories and concepts (Strauss and Corbin, 1998). All interview transcripts were analyzed separately to ensure that themes were in line with the associated context.

3.3.4. Model of higher-order themes

The generated themes were further explored and interpreted to create a model of higher-order themes (Appendix A). The higher-order themes were *prototyping and development, quality assurance,* and *enabling factors*. In addition, we identified patterns more general to the startup context. The 14 themes in the thematic map were extracted to address management of Agility (as shown in Table 5) and Quality (as shown in Table 6).

3.4. Validity procedure

In qualitative research, the validity must be addressed to enable replication of research and to ensure findings are trustworthy (Yin, 2003; Cruzes and Dybå, 2011; Runeson and Höst, 2009). To ensure

the validity of this study, we followed the validity guidelines from Runeson and Höst (2009).

Construct validity ensures that the operational measures that are studied really represent what the researcher have in mind and what is investigated according to the research questions. To assure that the interview questions (Section 3.2) were suitable for answering our research questions, we defined interview questions through a top-down approach using the Goal Question Metric method. Interviewees were either CEOs or engineers with insight into business- and technical-related aspects. Since it is difficult to understand a startup and its dimensions within a time-span of 30 to 60 min, we collected data about the startups through incubator and company websites prior to interviews. To improve the reliability of the study, all participating startups were included in the process of writing company descriptions to ensure their conformance with reality.

External validity refers to the extent to which the findings are generalizable beyond the context studied. For qualitative studies, the intention is to enable analytical generalization where the results are extended to companies which have common characteristics. Our startups were mostly located in Norway, mainly consisting of early-stage small-size entrepreneurial teams. They are also mostly self-funded and acquiring some key competence from the start. Hence, it would be safe to rely the findings to startups with similar characteristics (i.e., early-stage European startups). Startups from other American countries or startups already in a growth stage, might not be observed with similar features.

Reliability refers to the extent that data and the analysis are dependent on the specific researchers. We have defined and validated interview protocols with colleagues. To decrease the risk of biased interpretations, author one and two attended all interviews. Some interviews were in Norwegian, hence transcripts were not always verbatim to preserve the actual meaning of respondents. Recordings were transcribed shortly after each interview to mitigate bias. Additionally, we compared findings to related literature (Giardino et al., 2016; Nguyen-Duc et al., 2018; Ronkainen and Abrahamsson, 2003), examining similarities, contrasts, and explanations. Such comparisons have proven to enhance internal validity and the quality of findings (Eisenhardt, 1989).

4. Results - How do hardware startups achieve both agility and product quality during product development?

4.1. An integrative view on agility and product quality in hardware startup development

The integrative model of agility and quality in hardware startups is presented in Fig. 4. We have grouped the main concepts according to two dimensions (1) agility-driven or quality-driven, (2) project activities (i.e., prototype and product development, or quality assurance). Each concept describes a common foundation in hardware startups that manage agility or product quality. We classified the emerging concepts into three categories:

- Mindset (represented by green boxes in Fig. 4): a belief, an opinion, or a way of thinking towards a topic
- Practice (represented by pink boxes in Fig. 4): the actual application of an idea, a belief, or a method to solve a specific task
- Strategy (represented by yellow boxes in Fig. 4): a high level plan that might include a set of practices or processes to achieve a goal

As can be seen from Fig. 4, the integrative model of agility and quality in hardware startups focuses on four quadrants on two axes. The vertical axis shows two major activity areas (1) prototyping and product development and (2) quality assurance. The horizontal axis shows the area of Agility or Quality. By putting them together, we offer an integrative overview of how agility and quality are managed in both product development and quality assurance activities. The final section in the model represents enabling factors that apply to both quality and agility concepts. As seen from the model, hardware startups achieve agility at both mindset, strategy, and practice level in the prototyping and product development phase. Hardware startups include development practices during the quality assurance phase that provide short-term gains in quality. However, it becomes clear that hardware startups lack both strategies and mindsets for achieving the long-term quality of the product during the prototyping and development phases.

The model also illustrates the lack of practices during the quality assurance phase that support the vital need for agility in hardware startups. In other words, there are none quality-driven activities adopted by hardware startups that contribute to their agility. This impedes the adoption and focus on quality in hardware startups.

The commonality among hardware startups performing these approaches are (1) customized iterative practices, (2) sufficient competence in team, and (3) collaborative technical decision making. These appear as key mindsets and strategies for startups to perform both agility and quality-driven product development. In the following sub-sections, we present detailed insights related to the common enabling factors, agility and quality aspects in prototyping, product development, and quality assurance activities (Table 4).

4.2. Enabling factors for achieving agility and quality

Customized iterative practices. Hardware and hardware-oriented product development involve a lot of experimental work, and so developers are encouraged to follow an iterative development approach (Ronkainen and Abrahamsson, 2003). Among the startups, five practiced simplistic versions of Scrum, seven used ad-hoc Agile practices, while one startup did not follow a defined Agile development process. In some startups there was not identified a need to implement specific development methods, one reason being the small size of the development team. This was especially the case in early stages when tech teams were co-located and introduction of formal communication processes would inhibit the agility and freedom of the team. In the startup where the development team only consisted of one person, the degree of process was almost absent.

> S5 - *"Since the team is so small, communication is easy. We have not seen a need to implement any specific Agile methods or other lean practices."*

> S13 - *"I don't think Agile practices are applicable to hardware development, for example you cannot frequently re-design a port as it involves great costs."*

> S8 - *"In hardware, the variance of tasks and interrelated dependencies make it more complex than what current Scrum tools like Jira are suited for."*

> S4 - *"Strict Scrum is probably easier to implement for pure software development, so we use a simplified version of it."*

Due to different team sizes, product offerings, and other financial, managerial, and human factors, Agile practices were implemented differently among the hardware startups. Sprint duration usually lasted between 1–2 weeks, and goals were defined in weekly meetings. Since development of physical products usually takes longer time than implementation of software, the startups focused on defining measurable sub-goals that were part of a long-term plan. Most startups had the same Sprints for the respective hardware and software development. However, one startup differ-

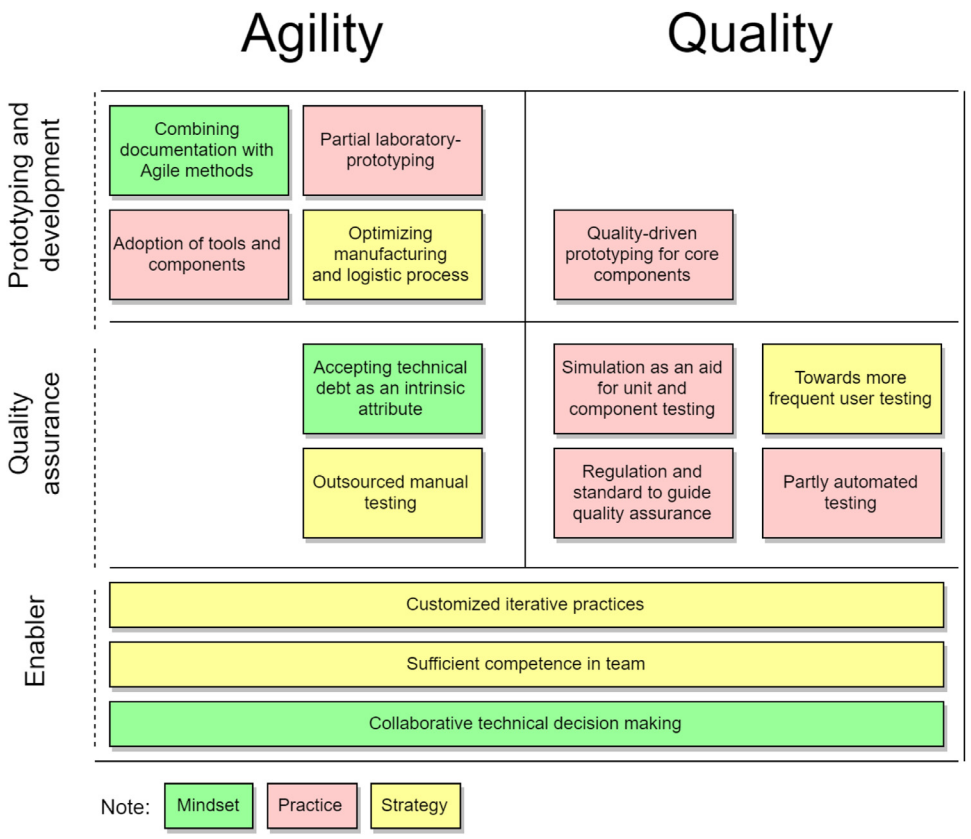


Fig. 4. An integrative view on quality and agility.

Table 4
Enabling factors for achieving agility and quality.

Terms	Definitions	Impacting factors	Category
Customized iterative practices	Self-defined versions of Agile with Sprints where customers or potential users can give feedback. Tailored set of Agile practices (e.g., product backlogs, Sprint reviews) might be adopted	Team competence, team size, third-party vendors, market feedback	Strategy
Sufficient competence in team	Acquiring in-house software and hardware engineers to perform design, implementation, and product testing	Funding, professional networks, recruitment strategy	Strategy
Collaborative technical decision making	Achieving harmony between hardware and software integration with a flat-team structure that supports quick decisions regarding both implementation and testing	Team competence, leadership, coordination and communication mechanisms	Mindset

Table 5
Achieving agility in hardware startups.

Terms	Definitions	Impacting factors	Category
Partial laboratory-prototyping	Production of simple and low-fidelity prototypes, representing a part of the final products and services, software and hardware prototyped separately	Complexity of hardware, funding, available tools, third-party vendors	Practice
Adoption of tools and components	Utilizing commercial-off-the-shelf or open source components and tools to speed up the prototyping	Funding, component-driven experts in team, third-party vendors	Practice
Optimizing manufacturing and logistic process	Managing third-party risks and maintaining flexibility in development process to achieve performance	Team competence, communication skills, risk management	Strategy
Combining documentation with Agile methods	Spend less time on documents, make it as a task in the Sprint backlog	Expert availability	Mindset
Accepting technical debt as an intrinsic attribute	Allow amount of technical debt that does not block product demonstration	Product nature, team competence, traceability of issues	Mindset
Outsourced manual testing	Outsourcing none important, manual testing tasks to third-party vendors	Communication skills, quality of outsourced partners, task definition mechanisms	Strategy

Table 6
Quality aspects of hardware startups.

Terms	Definitions	Impacting factors	Category
Quality-driven prototyping for core components	Test-driven prototyping, early focus on non-functional attributes	Testing capabilities within team	Practice
Towards more frequent user testing	Early verify customer value before thorough testing	Communication skills, task definition	Strategy
Partly automated testing	Team members individually test new functionality	Team competence, product nature	Practice
Simulation as an aid for unit and component testing	Ability to predict product behaviour before physical production	Available tools, product nature	Practice
Regulation and standard to guide quality assurance	Market regulations and standards may infer strict development guidelines	Quality of outsourced partners, available tools	Practice

entiated between hardware and software Sprints to better handle contingencies of hardware and software development.

S1 - *"We work on a weekly basis where we define goals for each week. These are part of a main goal of what we want to achieve during the semester."*

S10 - *"Software development follows two-week Sprints while hardware Sprints last 1–2 months."*

Sufficient competence in team. Although contracting is a common approach, startups mention that internal development would be the best way to achieve agility. Hardware startups need team members that are dedicated to all aspects of the development process. As hardware startups have to deal with many factors besides software there are higher demands to expertise and experience of team members. Team members of hardware startups will preferably need knowledge about application domain, systematic development, software and hardware development, mechanical engineering, and experience of working with third-party companies. Particularly, achieving a good collaboration between software engineers and hardware engineers in the team would accelerate the process of prototyping. However, this is only observed in one startup. Most of the startups had challenges of achieving right sets of competence from the beginning:

S6 - *"Finding talented people is hard. Since we are a startup we cannot give very good salary. This is why we try to attract people who see that the product may provide great value in the future."*

Even though external resources can substitute for the missing competence, this would not be sustainable in the long run. Many startups include part-time team members, who are typically more task-based oriented than co-founders. Depending on these people might reduce the agility of production due to the availability and commitment issues.

Collaborative technical decision making. Hardware startups are found with technology-driven processes of iterating their products. The teams are typically flat structured, probably due to the fact that startups often have a small number of members at early stages. Members are motivated and voluntary in taking tasks and responsibilities. In our study, startups seem to lack governance mechanisms of legal rules and strict regulation. Typically, technical decisions are made by engineers themselves. All decisions are made on team-basis. We also observe that startups allow for flexibility in working time and place, as everyone is responsible for the requirements needed for their area of responsibility. For a small team, team members could probably play multiple roles. Overall, team members trust other's competence. The team is flexible in organizing and reorganizing (i.e., adding new members and collaborating with vendors) to react to changes from environments.

4.3. Agility aspects of hardware startups

Partial laboratory-prototyping. Almost all startups immediately built a physical prototype to elicit requirements and achieve rapid business experimentation. They usually followed an evolutionary approach, performing incremental improvements on an early low-resolution prototype. Rapid prototyping is important to obtain customer feedback, however it can be problematic in the hardware context. Hardware startups usually have a significant focus on non-functional requirements because of the many challenges and regulations associated with complex systems development and the general hardware ecosystem. Common to the investigated startups is their priority of modularity both at software and hardware level, much so to achieve frequent user testing.

S10 - *"We made a physical prototype immediately. It looks like today's product, but with many shortcuts and lower quality."*

S8 - *"We can develop many low-resolution prototypes using our own equipment, but if we want high-quality prototypes we might have to order 10 different parts from 2 to 3 suppliers."*

To deal with their inability to quickly develop prototypes, the startups tried to be flexible on the software side of their products. Since software can be frequently updated and tested by customers, they focused on developing a simple interface between hardware and the software directly accessing the hardware. In this way they could achieve more parallel and independent development of hardware and software. They mainly tried to reuse software, as hardware components were easier to reuse with more refined prototypes.

S4 - *"We have developed a simple interface between hardware and software so that the development can happen individually."*

S3 - *"When we outsourced software development, changes took a lot of time... In software we need to make changes weekly. In hardware it is okay that things take a bit more time."*

S2 - *"We prefer making changes in the software or firmware. To facilitate this, we have a clearly defined interface between software and hardware."*

Adoption of tools and components. Among the investigated startups there was a more extensive reuse of software than hardware. Hardware and mechanical components were easier to reuse with more refined prototypes than early low-resolution prototypes. The startups made little use of mock-up tools, and so throwaway prototypes seem to take little part of the prototyping stage of hardware startups.

S2 - *"We try to reuse as much as possible from each prototype. We divide the code into different modules, so that if we replace any*

hardware component we only need to change that specific part of the code."

S7 - *"We tried to reuse the electronics, but it was harder than expected. So the physical components are usually substituted for each prototype... The software is more easily reusable."*

Having access to prototyping equipment will be an important asset, reducing both development time and prototyping cost. With 3D-printers startups can do a lot of the prototyping themselves, and make rapid changes based on customer feedback. This enables faster problem space testing.

S2 - *"With a 3D-printer we can make products that look and feel real. This is a huge advantage. We can literally do almost everything apart from the electronics production ourselves and put it together almost for free."*

S9 - *"We have done a lot of 3D-printing. Without access to useful equipment prototyping would have been very expensive and taken more time."*

Optimizing manufacturing and logistic process. The most time-consuming process of hardware prototyping is the long production and shipping times, as production usually is located in China or other countries in southeast Asia. This means that not only will the delivery time of necessary parts depend on the vendor's own schedule, but also the shipping method used. Several of the investigated startups spent a significant amount of money on speeding up production and shipping time of manufactured components.

S8 - *"All parts of the prototypes must be ordered, mostly from China, with long delivery times. We spend a lot of money making delivery times shorter."*

Several startups experienced quality issues working with their external partners. Manufacturing defects of crucial prototype components caused extra delays, which is critical considering the valuable time already spent waiting for the components. Cooperating with professional actors can decrease the risk of quality issues, and enhance communication.

S4 - *"We have outsourced production of mechanical parts and circuit boards. Some of the components we have received from the manufacturer have been in bad condition and with significant defects."*

As high-tech prototyping is a time demanding process, there might go several years from the startup is founded to the time a finalized product is ready to be released to the market. This implies that vendors' dependability also is of importance. Choosing components that with certainty will be available the entire prototyping stage is crucial.

S12 - *"The first version of the screen went out of production. This was the most important component and it took a lot of time to fix the problem."*

To achieve speed, product quality often gets low priority in startups. However, because of the vendor dependency of hardware startups, hardware development should receive higher focus on quality. Making shortcuts in hardware design, and not assuring that the design is of sufficient quality before sending the specifications for production might be costly. Initial findings suggest that hardware startups focus on ensuring the quality of core hardware components, as low-cost solutions may inhibit progress in the long-run. Findings from S12 and S1 indicate that hardware startups should put great effort into ensuring the quality of hardware components, as low-cost solutions can inhibit the long-term evolution of their prototypes.

S1 - *"We spent more than $500 on a single component we could not use. In addition we had to spend more time redesigning the board, and wait for it to be produced."*

Because of pressured financial resources and small production batches it can be hard for startups to receive commitment from professional manufacturers. Working with vendors producing components of high quality at an affordable cost will be an advantage. The big geographical distance, and the difference in language and culture may also challenge the communication skills of the team, as effective communication is important to receive service as paid for and maintain product development speed.

S10 - *"As we have grown, we have been able to work with better suppliers producing at higher quality, which in turn has helped us prototype faster."*

S2 - *"We are building on networks from earlier startup experience... Previously, we chose the cheapest suppliers, but then we also got components in bad condition, there were communication problems, and it usually took more than 4 weeks to get the products."*

Combining documentation with Agile methods. On the software side of the product, the common perception is that since the developers work on the code-base every day, documentation activities lead to additional overhead. Tacit knowledge seem to be a common practice in hardware startups.

S3 - *"We spend less time on documentation to speed things up, development is our main focus. It is also because software development is in-house. We work on it daily and understand the code."*

High-tech products include a lot of different sub-systems and technologies, and so product complexity increases fast. This implies that documentation of components should receive a bigger attention in hardware startups. In worst case, lack of quality and documentation can put all development on hold.

S2 - *"Instead of updating documentation and quality, we did things as fast as possible, which eventually led to a lot of extra work."*

The prototyping stage in hardware development is often significantly longer than that of software development. Since it might take years before hardware startups have a functioning product ready for the market, there's a great probability of people quitting the project before it is finished. As to this there should be increased focus on documentation in hardware startups, since knowledge often accompanies the person quitting.

S4 - *"Sometimes it becomes challenging to keep the knowledge of people who quit, the knowledge often accompanies that person. This leads to extra costs and effort."*

The choice of outsourcing companies can greatly impact the amount of documentation. Good partners usually provide well-documented solutions and components, which can help manage technical debt.

S3 - *"We received an 80-page user manual from the consultants who developed the hardware."*

To help startups perform documentation, there exist multiple tools lowering the barriers for writing documentation. Examples of tools include Wikis, Google Spreadsheets, and Confluence. Utilizing tools can help decrease the amount of rework in the long run. Also thorough documentation can allow for more efficient integration of new employees in the development process.

S2 - *"Previously we have spent a lot of extra time due to a lack of documentation. Instead of stopping, we did things as fast as pos-*

sible without performing documentation. This eventually lead to a lot of extra work."

S4 - *"We have a wiki for internal documentation. It is quite low effort to write something."*

Accepting technical debt as an intrinsic attribute. Technical debt has been illustrated by Brown et al. (2010), stating that "developers sometimes accept compromises in a system in one dimension (e.g., modularity) to meet an urgent demand in some other dimension (e.g., a deadline), and that such compromises incur a "debt" on which "interest" has to be paid and which the "principal" should be repaid at some point for the long-term health of the project". Finding the correct balance between learning goals and quality is therefore important in order to minimize waste and to manage technical debt (Terho et al., 2016).

By accepting that time to market is more important than product quality, hardware startups incur intentional technical debt. Business experimentation to build new features is performed in small iterative cycles with minimal effort on product quality to receive fast customer feedback. Corresponding to software startups technical debt mainly manifests itself on the software side of the product in hardware startups. Since software can be changed quickly, shortcuts and workarounds are more easily taken on the software side than on the hardware side of the product. The development team prioritizes implementing new functionality over improving the quality of the codebase.

S2 - *"Software changes all the time... To make things work straight away, we'd rather take a shortcut and fix it later. We know we're building up technical debt, but it's on purpose to be able to test the product on customers as quickly as possible."*

Hardware startups do not accumulate technical debt for their hardware components similarly as for their software components. As the concept of technical debt is built on erosion of systems from frequent low-quality changes, this is not as easily manifested in hardware components. Refactoring delivered or released hardware is a difficult and rarely performed endeavour. However, poor hardware design might eventually lead to the hardware needing to be redesigned. As hardware components often are reused between low-resolution prototypes, bad design might imply that the hardware needs redesign on an earlier iteration of the product than intended. Early lifetime design decisions might propagate throughout the lifetime of the product, and may eventually become part of the final product. These poorly made design decisions will then be discovered after the product is released. Hence, temporary low-quality solutions in both hardware and software will eventually lead to accumulation of technical debt in hardware startups

Outsourced manual testing. Outsourcing includes the choices of both local consultant companies and aboard contracting vendors. Hardware development requires a significant amount of testing to ensure product quality. This applies already at the prototype stage, and for demonstration. Among the companies, some outsourced their manual testing (i.e., testing the final release at different execution environments, and testing the integration between developed components and known services or products). Outsourcing manual testing can save time and effort for startups to focus on innovation and core value creation activities.

S10 - *"In software we have a great focus on testing. When software is modified, we run automatic tests to ensure that everything works... In hardware we test that the product functions in different climates, and perform various mechanical tests... We have also outsourced much manual testing to a company to check more parts of the product."*

4.4. Quality aspect of hardware startups

Quality-driven prototyping for core components. Testing is central to hardware startups. High quality in hardware development is important both because of the cost associated with mistakes from production, but also as quality greatly affects the perceived functionality of the product. In contrast to software products, it is challenging to implement changes and make improvements to the quality after the product has been produced and assembled. As a consequence, focus on non-functional attributes at the prototyping stage is essential. We observed many startups that implemented a test-driven approach for developing the core components of their prototypes.

S4 - "We have a test setup to ensure that the subsystems work as intended, and that allows us to analyze different metrics and data. For the most critical components and features we usually define detailed test plans in advance of development."

Towards more frequent user testing. To achieve quick development speed in early stages, low-level testing activities generally receive little focus in hardware startups. Before a feature is guaranteed to be part of the final product, it is more important to verify that the feature adds value to the customers. Until then, the time spent on testing activities is minimized. This is also evident in software startups, where developers avoid wasting time on improvements of not-validated functionalities (Giardino et al., 2016).

S2 - *"We prefer to work fast, as writing tests can double the development time... If parts are to be replaced, then we think there's no point in spending time on testing."*

In S3 and S6, the CEO highlighted the importance of having frequent feedback from their customers and users on the prototypes. This would be critical for the design and development of a product that later is sought to a mass market.

Several startups faced the challenge of testing their product in realistic environments because of legal restrictions related to privacy and public safety. Simulations and dummy-data can be alternatives to early testing.

S4 - *"Setting up a foundation for doing robust tests is a challenge. When developing drones it is not easy to perform testing, it requires specific experience and knowledge."*

Lack of financial resources and long delivery times make it challenging to test the product on a broader spectrum of customers. Physical prototypes are resource-intensive to develop, and in contrast to pure software products, one cannot necessarily deliver a new digital software update to customers. The investigated startups relied on a small set of customers for frequent feedback.

Partly automated testing. The hardware startups relied on each individual developer to test features as they were implemented. In that way the person responsible for the code was also responsible for its quality and functioning with the rest of the system. A frequently used testing activity among the startups was manual smoke tests (i.e., ensuring that major functionality work before undertaking more formal testing procedures). Prototypes were manually tested by internal employees to identify the most prominent defects before testing prototypes with early adopters or customers.

S8 - *"We test the subsystems ourselves, but do not have a structured system for testing... The person responsible for delivery is also responsible for testing the feature to make sure it works."*

S1 - *"People inside the startup who have experience with similar solutions test the product before it is tested with pilot customers."*

Software engineers tends to optimize the integration and operation of software components by adopting automated testing. This is

reported to be done in some part of the product. *Simulation as an aid for unit and component testing.* For hardware development, simulation is very helpful to ensure certain quality attributes of physical products. Hardware simulations can help determine the precise operation of hardware without producing an expensive prototype, and enable testing of the hardware-software co-operation (Ronkainen et al., 2002). Several startups found it challenging to test their product in realistic environments, both due to memory and performance constraints and because of privacy and public safety issues. Since planning is difficult in the startup context, test plans were often changed, hence these were often neglected. Simulations helped testing the product and code base before production, postponing the split between hardware and software functionality.

> S4 - *"At an early stage, things don't always go as planned. Other things than what you test for fail, so test and project plans often change a lot... In addition to performing many simulations, we use basic tuning of attitude control to avoid simple mathematical errors."*

Among the investigated startups we found that startups in later lifecycle stages implemented more systematic testing activities. As they got more customers, quality and testing activities became more important. Established customers have stricter demands than pilot customers. To deal with increased quality requirements, the startups implemented formal processes for testing. *Regulation as a guide for early quality assurance.* For some startups working under regulated domains, such as healthcare and automotive, market regulations and standards infer a strict guideline for product development. This has been used to guide the quality assurance activities since the prototyping phases. Besides, companies operating in the market will need to document all parts of their product and meet the high standards of quality required. Hence, market segment will greatly affect the severity of technical debt and infers an early debt management.

> S6 - *"We are weak on processes and document management, it is very ad-hoc. Soon we will introduce a process tool and a document management tool. This is necessary if we are to meet the ISO standard requirements and get it approved as a medical product."*

> S13 - *"The documentation is part of the development process... We have an ISO certification that says we are certified according to that quality process. They have strict requirements on how documentation should be kept, including the flow of the documentation and what kind of documentation to write."*

5. Discussion

5.1. Achieving agility in hardware-related product development

Agility is an essential part of startups in general (Pantiuchina et al., 2017), and should thus be considered as an attribute of early stage hardware startups. Hardware startups' need for speed often sacrifice product quality. Instead of applying best engineering principles, we found that development teams prefer simple solutions to achieve rapid business growth. Speed-related activities lead to the accumulation of technical debt, which eventually inhibit further business growth (Giardino et al., 2016). Achieving agility in hardware startups is not as straightforward as adopting Agile practices or rapid prototyping in software startups.

It is evident that iterative development with middle-term planning is used in hardware startups because hardware development usually requires more time than software development does. As non-functional attributes need to be assured at the prototyping stage (Nguyen-Duc et al., 2018), and hardware startups deal with third-party dependency, release frequency is low compared to software startups. This limited the ability of continuous experimentation as observed in software startups (Fagerholm et al., 2014).

The investigated hardware startups achieved agility by facilitating for simultaneous work on multiple possible solutions. Implementation of ready-made or outsourced components can be a significant struggle as hardware startups rarely develop all components themselves. System design and architectural decisions are made in advance of development, and may greatly affect later system integration of components. As development in hardware startups can be considered a test of feasibility, development methods should facilitate for experimentation of multiple solution methods.

One of the key practices to achieve agility is efficient management of external dependency. By increasing the knowledge of external components in the system, developing reliable relationships with external partners, startups can reduce the time wasted on fixing issues that are not under the startup's team control.

The nature of hardware development makes embedded systems sensitive to rapid changes in hardware or hardware-related software (Ronkainen and Abrahamsson, 2003). Preliminary architecture design is necessary to facilitate iterative development, and flexibility to handle rapid changes. As hardware startups intentionally try to force changes on the software side, neglecting up-front design may cause bugs that are not easily detected. Early investments in up-front system design can make the product more robust to changes, and facilitate for streamlined development in later stages.

Testing must ensure conformance between hardware and hardware-related software. However, the test-driven approach is problematic because of the severe memory and performance constraints of embedded systems (Ronkainen and Abrahamsson, 2003), in addition to the restricted resources of hardware startups. To achieve quick development speed in early stages, low-level testing activities generally receive little focus in hardware startups. The startups were first and foremost interested in ensuring that included features provide value to customers.

Refactoring is basically the object-oriented variant of restructuring, "the process of changing a [object-oriented] software system in such a way that it does not alter the external behaviour of the code, yet improves its internal structure" (Fowler, 2018). Our research indicates that regular refactoring is not practiced in hardware startups, neither for software or hardware development. Prototyping consists to a large degree of shortcuts and workarounds, especially for the software components. The nature of hardware development is not compatible with regular refactoring, as frequently redesigning components involves significant costs. This relates to software startups as well. Research states that refactoring rarely is implemented in the early stages of the startup, but as the startup grows, returning the accumulated technical debt is needed to meet more quality-demanding customers and scalability issues (Giardino et al., 2016).

The mentioned practices extend the list of Agile methods for embedded systems development and in general hardware-related products (i.e., embedded hardware, wearable devices, Internet-of-things systems, and robotics) (Kaisti et al., 2013). The environmental conditions that make the practices particularly relevant include limited resources, market-driven requirements, and the temporary and exploratory nature of process management.

5.2. Assuring product quality

The complexities and uniqueness of hardware development imply that hardware startups need to prioritize product quality differently from software startups in order to speed-up development. The investigated startups tried to facilitate for changes in their software parts while keeping the amount of hardware rework minimum, due to the rigid nature of hardware development. Hardware

quality is often necessary to meet real-time performance requirements of embedded systems (Ronkainen and Abrahamsson, 2003). Enabling the hardware-software co-operation is an intricate process due to the complex control and testing support required over hardware, and the fast time-to-market cycles require simultaneous software and hardware design (Ronkainen et al., 2002). The hardware startups invested in a simple interface combined with a skilled team to increase the amount of parallel development, facilitating for two largely independent development processes of hardware and software.

While forcing rapid changes on the flexible software side, the hardware startups incurred intentional technical debt. Since the software developers constantly worked with the code base, they relied on tacit knowledge instead of formal documentation. Hardware documentation seemed to be of higher importance due to the many stakeholders involved in hardware development. Intentional technical debt is a frequent problem in software startups, but can be even more harmful for hardware startups due to the change-sensitivity of the numerous complex hardware-software interactions (Ronkainen and Abrahamsson, 2003). Refactoring of code base can cause changes in system behaviour or even jeopardize system operation. Even if software shortcuts make sense in the short-run, our findings indicate that the complex nature of high-tech products may cause a severe amount of rework in the long-run. Hardware startups should invest in documentation tools to lower the barriers for formal documentation. Adoption of Agile methods has proven to be efficient in reducing error rates (Albuquerque et al., 2012), however current usage of such is restricted to a subset of Agile practices customized the individual needs of hardware startups.

The investigated hardware startups incurred unintentional technical debt due to the difficulty of testing problem space. They performed usability and acceptance tests on a small group of pilot customers, as a lack of financial resources and third-party dependencies (e.g., delivery times) made it challenging to test the product on a broader spectre of customers. By immediately building a physical prototype, the startups focused on validating, as they focused on making their customer acquisition processes more efficient rather than testing the demand for a functional product. The hardware startups' inability to produce many prototypes inhibited business experimentation and lead to feature creep. Feature creep in hardware startups may similarly to software startups be harmful to the production and maintenance of core functionality (Nguyen-Duc et al., 2017).

Testing is central to embedded system development, as hardware startups need to assure non-functional attributes at an early stage. We found that testing practices were implemented to various extent among the hardware startups, among other things, because the testing environment was different from the development environment. Memory and performance constraints can also affect hardware startups' testing ability (Ronkainen and Abrahamsson, 2003). The investigated startups relied on individual developers' efforts to ensure quality of new functionality. Manual smoke tests and simulations were favored to professional engineering activities. Findings indicate that rigorous low-level testing practices were not implemented before later life-cycle stages.

The investigated startups followed a quality-driven development approach, where performance and quality criteria of core components were verified through frequent user testing. Beyond functional testing as in software development, specific test plans are needed for hardware and hardware-software integration interfaces. The found practices can be applied to a cost and quality-driven environment similar to what (Peters et al., 2014) reported.

5.3. Balancing agility and quality in high-tech product development

The conflicting requirements for quality and agility mean development methods will need a hybrid process that balances both strict hardware development while allowing speed and flexibility as in software development. We extend knowledge about possible integrative approaches for agility and quality in hardware development (Jha et al., 2016). Tactics for achieving agility (i.e., outsourcing, rapid prototyping, Sprint-based development) related to speed are commonly used by most startups, however, we see that hardware startups' overall strategy is to spend more time on quality-assuring activities.

A previous study reports five types of Agile practices that influence software quality, which are teamwork, engineering, documentation, testing, and management (Arcos-Medina and Mauricio, 2019). The startups in our study illustrate the implementation of simple quality-aware practices in their development process with the focus on frequent user testing, early customer feedback, collaborative decision making, adoption of low-level documentation tools, and model-based engineering.

Working with limited resources, finding alternative ways to ensure product quality in early stages can be of high value. Realistic testing environments may be restricted, and as the early stages should not only be about failing fast, but failing cheap, computer simulations may provide early product validation. As documentation and component testing usually is the responsibility of each developer it should be easy to produce light documentation. Finding a sufficient approach to Agile documentation in startups that does not disrupt the informal workflow of the team is important. Simple and useful documentation will spare later effort.

Particular for the startup context, hardware and software teams are not always co-located or communicating in an effective manner. We see hardware-software integration meetings as an important practice for providing agility and quality to the development process, supporting team decision making. As observed in most startups, managing the interface between hardware and software is a necessity for speed that allows for distributed development teams simultaneously working on multiple solutions and technologies.

Existing research addresses the combination between agility and quality at requirement engineering, architecture, and implementation level (Jha et al., 2016; Arcos-Medina and Mauricio, 2019; Franch et al., 2019). Our study offers a comprehensive view on adopting agility and quality-aware practices across product development activities. We also observe that all startups to some extent were familiar with Agile and it's concepts, however its' applicability to the hardware startup context were of different perception. Although quality-aware Agile practices are useful, there is still a lack of know-how to establish and bring these practices into actual usage. Hardware startups need a specific set of quality-aware practices in order to manage technical debt and attain the level of quality required for all stages of their development process.

6. Conclusion

Hardware startups develop physical products with mixed hardware and software components, requiring expertise within a broad range of technological fields. In addition to software development hardware startups deal with production and logistics issues, factors implying higher initial financial and human investments than what is experienced by software startups. From a qualitative exploratory study investigating 13 hardware startups, this paper presents the role of engineering activities from idea conceptualization to a launched product, and factors influencing agility and quality. Our

research results indicate that hardware startups achieve rapid prototyping through evolutionary approaches, hardware-software decomposition strategies, and opportunistic Agile practices. The level of agility in prototyping and production varies depending on team competence, funding and various external constraints. Hardware startups incur technical debt as an unavoidable part of the evolution. The state-of-practice testing, with informal and partial quality assurance approaches, does not help to reduce the overall level of technical debt. The competitive environment of hardware startups makes speed inevitable, where investing in hardware quality will be necessary for bringing products fast to market. The study explains the priorities of hardware startups' engineering approach, and the specific need for process in managing the relationship between quality and speed. We provide practitioners with a better understanding and awareness of their own context, helping them make technical and business-related decisions of sustainable character. It is also evident that quality and agility is balanced with the mean of quality-aware Agile processes with an effective management of third-party vendors.

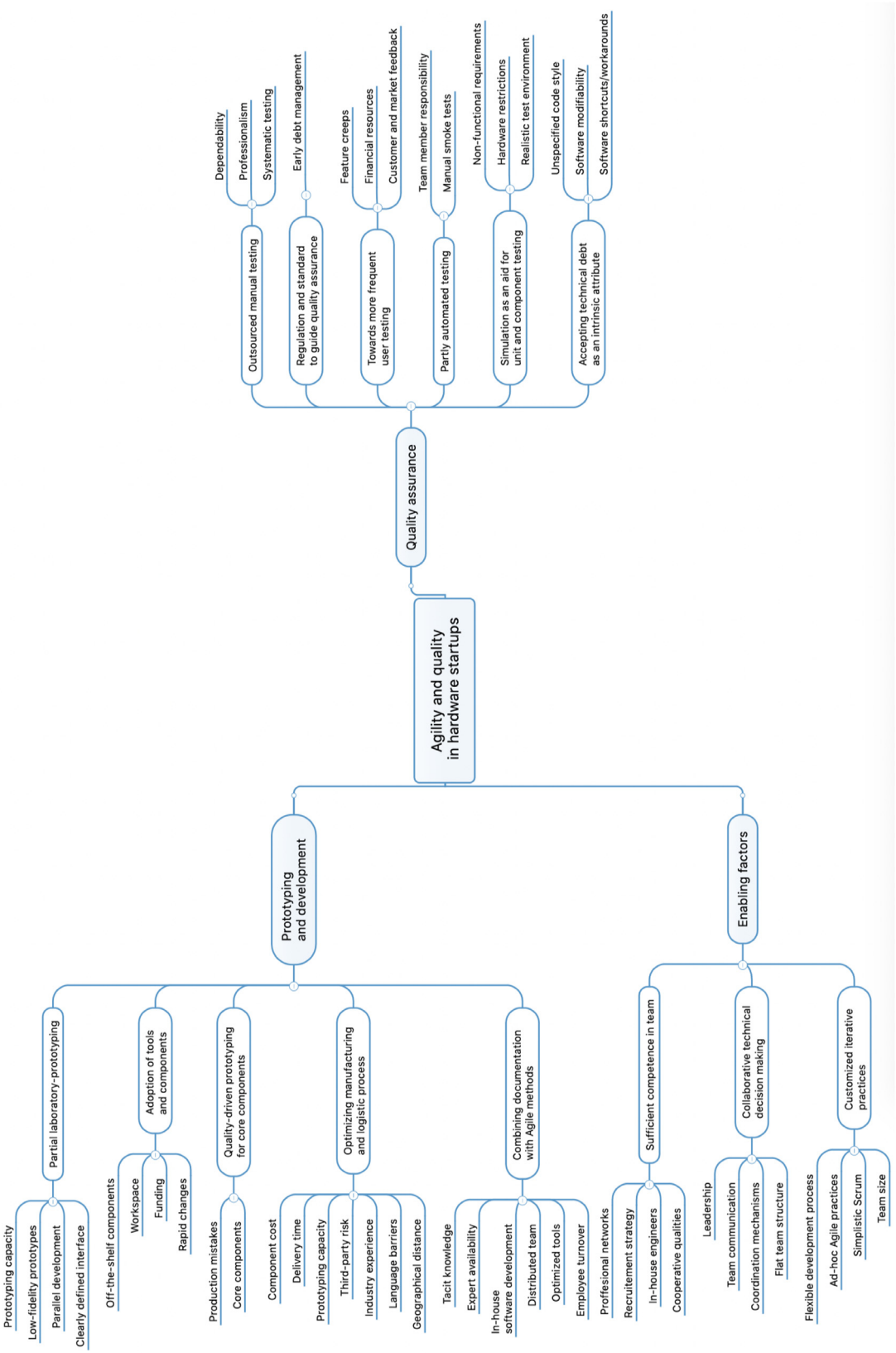
This study provides early empirical evidence into prototyping and engineering practices in hardware startups. However, the study highlights the compromise hardware startups make between quality and speed. Quality is of higher significance, and more research should be provided identifying valuable activities and approaches for hardware startups dealing with restricted resources. We encourage researchers to explore the long-term effects of technical debt, as our results are based on a small sample of early-stage hardware startups. In addition, future research should investigate how hardware startups can ensure safety and security standards when developing highly safe systems, following standards like IEC61508 (Japan, 2012). The results are partly based on managerial viewpoints, hence missing important links to everyday testing activities performed by engineers and developers. Future work should verify the results with other startup companies to find its applicability in other environments, enabling generalization to a larger startup audience. More investigations should be undertaken to understand the role of scope in the engineering activities of hardware startups.

Our integrative model of agility and quality also implies the focus on mindsets, strategies, and practices for each product development activity. Future research should focus on defining quality-driven practices that contribute to agility, and further simplify the introduction of quality in startups. As hardware startups need more attention to hardware quality to allow for evolutionary prototyping and speed, there should be engineering strategies describing how hardware startups can manage the relationship between restricted resources and increased quality demands. Future research can also focus on strategies and mindsets to support long-term quality in the startup context. Hardware startups need specific guidelines for performing problem space testing, and research should verify the consequences of its absence. There are identified several limitations to this study. Having based our study on qualitative measures, results and implications are subject to bias. To mitigate the risk of misunderstandings or wrong interpretations, two researchers attended all interviews. Whenever possible, interviews were performed face-to-face on-site. Recordings were transcribed and translated shortly after each interview to ensure respondents' meanings were preserved. Another limitation is the insufficient knowledge on technical decisions and product development challenges provided by some interviewees (i.e., knowledge of business executives is often based on managerial viewpoints). The results would benefit from a greater amount of participants providing insights into every-day engineering activities of hardware startups. Another shortcoming to the study is the diversity of the investigated startups, as the selection constituted early-stage European hardware startups. The study would profit from a wider collection of data, both to discover more relevant themes and to ensure credible conclusions (i.e., generalizability of the results). Further investigations of hardware startups operating in different markets, lifecycle stages, and various geographical locations can improve the reliability of the research results.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Thematic map



Appendix B. Pre-interview question questionnaire

- Briefly describe your product.
- Briefly explain your role and responsibilities in the company.
- Briefly describe your company (i.e., history, current headcount, roles, and process)
- Have you received any funding?

CRediT authorship contribution statement

Vebjørn Berg: Conceptualization, Methodology, Formal analysis, Investigation, Data curation, Writing - original draft, Writing - review & editing. **Jørgen Birkeland:** Conceptualization, Methodology, Formal analysis, Investigation, Data curation, Writing - original draft, Writing - review & editing. **Anh Nguyen-Duc:** Conceptualization, Validation, Resources, Supervision, Writing - original draft, Writing - review & editing. **Ilias O. Pappas:** Resources, Supervision, Writing - review & editing. **Letizia Jaccheri:** Validation, Resources, Supervision, Writing - review & editing.

References

Abatecola, G., Cafferata, R., Poggesi, S., 2012. Arthur stinchcombe's "liability of newness": contribution and impact of the construct. J. Manage. Hist. 18 (4), 402–418. doi:10.1108/17511341211258747.

Abrahamsson, P., Oza, N., Siponen, M.T., 2010. Agile software development methods: a comparative review. In: Agile Software Development. Springer, pp. 31–59.

Cunningham et al., W., 2001. The agile manifesto. http://www.agilemanifesto.org access date: 2017-11-12.

Albuquerque, C.O., Antonino, P.O., Nakagawa, E.Y., 2012. An investigation into agile methods in embedded systems development. In: International Conference on Computational Science and Its Applications. Springer, pp. 576–591.

Aldrich, H., Auster, E.R., 1986. Even dwarfs started small: liabilities of age and size and their strategic implications.. Res. Organ. Behav..

Alvarez, S.A., Barney, J.B., 2007. Discovery and creation: alternative theories of entrepreneurial action. Strateg. Entrep. J. 1 (1–2), 11–26.

Alves, C., Pereira, S., Castro, J., 2006. A study in market-driven requirements engineering.

Andersson, C., Runeson, P., 2002. Verification and validation in industry - a qualitative survey on the state of practice. In: Proceedings International Symposium on Empirical Software Engineering, pp. 37–47. doi:10.1109/ISESE.2002.1166923.

Arcos-Medina, G., Mauricio, D., 2019. Aspects of software quality applied to the process of agile software development: a systematic literature review. Int. J. Syst. Assur. Eng. Manage. 10 (5), 867–897. doi:10.1007/s13198-019-00840-7.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D., 2001. Manifesto for agile software development.

Berg, V., Birkeland, J., Nguyen-Duc, A., Pappas, I., Jaccheri, L., 2018. Software startup engineering: a systematic mapping study. J. Syst. Softw..

Blank, S., 2013a. The Four Steps to the Epiphany: Successful Strategies for Products That Win. BookBaby.

Blank, S., 2013b. Why the lean start-up changes everything. Harv. Bus. Rev. 91 (5), 63–72.

Bohmer, B.A.A.I., Lindemann, U., 2015. Open innovation ecosystem-makerspaces within an agile innovation process. In: Proceedings of the International Society for Professional Innovation Management.

Bosch, J., 2016. Speed, data, and ecosystems: the future of software engineering. IEEE Softw. 33 (1), 82–88. doi:10.1109/MS.2016.14.

Brown, N., Cai, Y., Guo, Y., Kazman, R., Kim, M., Kruchten, P., Lim, E., MacCormack, A., Nord, R., Ozkaya, I., 2010. Managing technical debt in software-reliant systems. In: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research. ACM, pp. 47–52.

Carmel, E., 1994. Time-to-completion in software package startups. In: 1994 Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences.

Coleman, G., O'Connor, R.V., 2008. An investigation into software development process formation in software start-ups. J. Enterp. Inform. Manage. 21 (6), 633–648. doi:10.1108/17410390810911221.

Conboy, K., 2009. Agility from first principles: reconstructing the concept of agility in information systems development. Inform. Syst. Res. 20 (3), 329–354.

Crowne, M., 2002. Why software product startups fail and what to do about it. Evolution of software product development in startup companies. In: Engineering Management Conference, 2002. IEMC'02. 2002 IEEE International. IEEE, pp. 338–343.

Cruzes, D.S., Dybå, T., 2011. Recommended steps for thematic synthesis in software engineering. In: Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on. IEEE, pp. 275–284.

Da Silva, T.S., Martin, A., Maurer, F., Silveira, M., 2011. User-centered design and agile methods: a systematic review. In: 2011 Agile Conference. IEEE, pp. 77–86.

Dahlstedt, A., 2003. Study of current practices in market-driven requirements engineering. In: Third Conference for the Promotion of Research in IT at New Universities and University Colleges in Sweden.

Díaz, J., Pérez, J., Alarcón, P.P., Garbajosa, J., 2011. Agile product line engineering—a systematic literature review. Software 41 (8), 921–941.

DiResta, R., Forrest, B., Vinyard, R., 2015. The Hardware Startup: Building Your Product, Business, and Brand. O'Reilly Media, Inc..

Dybå, T., Dingsøyr, T., 2008. Empirical studies of agile software development: asystematic review. Inf. Softw. Technol. 50 (9–10), 833–859.

Eisenhardt, K.M., 1989. Building theories from case study research. Acad. Manage. Rev. 14 (4), 532–550.

Fagerholm, F., Guinea, A.S., Mäenpää, H., Münch, J., 2014. Building blocks for continuous experimentation. In: Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering. ACM, pp. 26–35.

Fowler, M., 2018. Refactoring: Improving the Design of Existing Code. Addison-Wesley Professional.

Franch, X., Lopez, L., Martínez-Fernández, S., Oriol, M., Rodríguez, P., Trendowicz, A., 2019. Quality-aware rapid software development project: the q-rapids project. In: Software Technology: Methods and Tools. Springer, Cham, pp. 378–392. doi:10.1007/978-3-030-29852-4_32.

Garbajosa, J., Magnusson, M., Wang, X., 2017. Generating innovations for the internet of things: agility and speed. In: Proceedings of the XP2017 Scientific Workshops. ACM, p. 10.

Ghanam, Y., Maurer, F., 2010. Linking feature models to code artifacts using executable acceptance tests. In: International Conference on Software Product Lines. Springer, pp. 211–225.

Ghezzi, A., 2018. Digital startups and the adoption and implementation of lean startup approaches: effectuation, bricolage and opportunity creation in practice. Technol. Forecast Soc. Change doi:10.1016/j.techfore.2018.09.017.

Giardino, C., Bajwa, S.S., Wang, X., Abrahamsson, P., 2015. Key challenges in early-stage software startups. In: International Conference on Agile Software Development. Springer, pp. 52–63.

Giardino, C., Paternoster, N., Unterkalmsteiner, M., Gorschek, T., Abrahamsson, P., 2016. Software development in startup companies: the greenfield startup model. IEEE Trans. Softw. Eng. 42 (6), 585–604. doi:10.1109/TSE.2015.2509970.

Giardino, C., Unterkalmsteiner, M., Paternoster, N., Gorschek, T., Abrahamsson, P., 2014a. What do we know about software development in startups? IEEE Softw. 31 (5), 28–32. doi:10.1109/MS.2014.129.

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

Hanssen, G.K., Fægri, T.E., 2008. Process fusion: an industrial case study on agile software product line engineering. J. Syst. Softw. 81 (6), 843–854.

Hodgson, G.M., Knudsen, T., 2004. The firm as an interactor: firms as vehicles for habits and routines. J. Evol. Econ. 14 (3), 281–307. doi:10.1007/s00191-004-0192-1.

Jacobson, I., Spence, I., Ng, P.-W., 2017. Is there a single method for the internet of things? ACM Queue 15 (3), 20.

Jalali, S., Wohlin, C., 2010. Agile practices in global software engineering-a systematic map. In: 2010 5th IEEE International Conference on Global Software Engineering. IEEE, pp. 45–54.

Japan, I., 2012. Embedded System Development Process Reference Guide. Information-technology Promotion Agency, Japan.

Jha, M.M., Vilardell, R.M.F., Narayan, J., 2016. Scaling agile scrum software development: providing agility and quality to platform development by reducing time to market. In: 2016 IEEE 11th International Conference on Global Software Engineering (ICGSE), pp. 84–88 ISSN: 2329-6313. doi:10.1109/ICGSE.2016.24.

Kaisti, M., Rantala, V., Mujunen, T., Hyrynsalmi, S., Könnölä, K., Mäkilä, T., Lehtonen, T., 2013. Agile methods for embedded systems development-a literature review and a mapping study. EURASIP J. Embedded Syst. 2013 (1), 15.

Karlsson, L., Dahlstedt, A., och Dag, J.N., Regnell, B., Persson, A., 2002. Challenges in market-driven requirements engineering-an industrial interview study. Eighth International Workshop on Requirements Engineering: Foundation for Software Quality.

Keil, M., Carmel, E., 1995. Customer-developer links in software development. Commun. ACM 38 (5), 33–44.

Klein, H.K., Myers, M.D., 1999. A set of principles for conducting and evaluating interpretive field studies in information systems. MIS Q. 23 (1), 67–93. doi:10.2307/249410.

Klotins, E., Unterkalmsteiner, M., Gorschek, T., 2015. Software engineering knowledge areas in startup companies: a mapping study. In: Lecture Notes in Business Information Processing, vol. 210, pp. 245–257.

Marmer, M., Herrmann, B. L., Dogrultan, E., Berman, R., Eesley, C., Blank, S., 2011. Startup Genome Report Extra: premature Scaling. Startup Genome, 10.

Misra, S., Kumar, V., Kumar, U., Fantazy, K., Akhter, M., 2012. Agile software development practices: evolution, principles, and criticisms. Int. J. Qual. Reliab.Manage. 29 (9), 972–980.

Nguyen-Duc, A., Abrahamsson, P., 2016. Minimum viable product or multiple facet product? The role of MVP in software startups. In: International Conference on Agile Software Development. Springer, pp. 118–130.

Nguyen-Duc, A., Seppänen, P., Abrahamsson, P., 2015. Hunter-gatherer cycle: a conceptual model of the evolution of software startups. In: Proceedings of the 2015 International Conference on Software and System Process. ACM, New York, NY, USA, pp. 199–203 Event-place: Tallinn, Estonia. doi:10.1145/2785592.2795368.

Nguyen-Duc, A., Shah, S.M.A., Ambrahamsson, P., 2016. Towards an early stage software startups evolution model. In: Software Engineering and Advanced Applications (SEAA), 2016 42th Euromicro Conference on. IEEE, pp. 120–127.

Nguyen-Duc, A., Wang, X., Abrahamsson, P., 2017. What influences the speed of prototyping? An empirical investigation of twenty software startups. In: International Conference on Agile Software Development. Springer, pp. 20–36.

Nguyen-Duc, A., Weng, X., Abrahamsson, P., 2018. A preliminary study of agility in business and production: cases of early-stage hardware startups. In: Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. ACM, p. 51.

Oates, B.J., 2005. Researching Information Systems and Computing. Sage.

Pantiuchina, J., Mondini, M., Khanna, D., Wang, X., Abrahamsson, P., 2017. Are software startups applying agile practices? The state of the practice from a large

survey. In: 18th International Conference on Agile Software Development, XP 2017, May 22, 2017 - May 26, 2017. Springer Verlag, pp. 167–183. doi:10.1007/978-3-319-57633-6_11.
Paternoster, N., Giardino, C., Unterkalmsteiner, M., Gorschek, T., Abrahamsson, P., 2014. Software development in startup companies: asystematic mapping study. Inform. Softw. Technol. 56 (10), 1200–1218. doi:10.1016/j.infsof.2014.04.014.
Peters, H., Knieke, C., Brox, O., Jauns-Seyfried, S., Krämer, M., Schulze, A., 2014. A test-driven approach for model-based development of powertrain functions. In: Cantone, G., Marchesi, M. (Eds.), Agile Processes in Software Engineering and Extreme Programming. Springer International Publishing, pp. 294–301. doi:10.1007/978-3-319-06862-6_23.
Rafiq, U., Bajwa, S.S., Xiaofeng, W., Lunesu, I., 2017. Requirements elicitation techniques applied in software startups. In: 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 30 Aug.-1 Sept. 2017. IEEE Computer Society, pp. 141–144. doi:10.1109/SEAA.2017.73.
Ries, E., 2011. The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Crown Books.
Robson, C., 2002. Real World Research: A Resource for Social Scientists and Practitioner-Researchers, second ed. Wiley-Blackwell, Oxford, UK ; Madden, Mass.
Ronkainen, J., Abrahamsson, P., 2003. Software development under stringent hardware constraints: do agile methods have a chance? In: International Conference on Extreme Programming and Agile Processes in Software Engineering. Springer, pp. 73–79.
Ronkainen, J., Taramaa, J., Savuoja, A., 2002. Characteristics of process improvement of hardware-related sw. In: International Conference on Product Focused Software Process Improvement. Springer, pp. 247–257.
Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empir. softw. Eng. 14 (2), 131.
Saldaña, J., 2015. The Coding Manual for Qualitative Researchers. Sage.
Sarasvathy, S.D., 2001. Causation and effectuation: toward a theoretical shift from economic inevitability to entrepreneurial contingency. Acad. Manage. Rev. 26 (2), 243–263.
Sommer, S.C., Loch, C.H., Dong, J., 2009. Managing complexity and unforeseeable uncertainty in startup companies: an empirical study. Organ. Sci. 20 (1), 118–133.
Steininger, D.M., 2019. Linking information systems and entrepreneurship: areview and agenda for IT-associated and digital entrepreneurship research. Inform. Syst. J. 29 (2), 363–407. doi:10.1111/isj.12206.
Stinchcombe, A.L., 2000. Social structure and organizations. In: Economics Meets Sociology in Strategic Management. Emerald Group Publishing Limited, pp. 229–259. doi:10.1016/S0742-3322(00)17019-6.
Strauss, A., Corbin, J., 1998. Basics of qualitative research: procedures and techniques for developing grounded theory.
Sutton Jr., S.M., 2000. Role of process in a software start-up. IEEE Softw. 17 (4), 33–39. doi:10.1109/52.854066.
Terho, H., Suonsyrja, S., Systa, K., 2016. The developers dilemma: perfect product development or fast business validation? In: 17th International Conference on Product-Focused Software Process Improvement, PROFES 2016, November 24, 2016 - November 26, 2016. Springer Verlag, pp. 571–579. doi:10.1007/978-3-319-49094-6_42.
Tripathi, N., Annanpera, E., Oivo, M., Liukkunen, K., 2016. Exploring processes in small software companies: a systematic review. In: Communications in Computer and Information Science, vol. 609, pp. 150–165.
Unterkalmsteiner, M., Abrahamsson, P., Wang, X.F., Anh, N.D., Shah, S., Bajwa, S.S., Baltes, G.H., Conboy, K., Cullina, E., Dennehy, D., Edison, H., Fernandez-Sanchez, C., Garbajosa, J., Gorschek, T., Klotins, E., Hokkanen, L., Kon, F., Lunesu, I., Marchesi, M., Morgan, L., Oivo, M., Selig, C., Seppanen, P., Sweetman, R., Tyrvainen, P., Ungerer, C., Yague, A., 2016. Software startups - a research agenda. E-Inform. Softw. Eng. J. 10 (1), 89–123. doi:10.5277/e-Inf160105.
Van Gelderen, M., Thurik, R., Bosma, N., 2005. Success and risk factors in the pre-startup phase. Small Bus. Econ. 24 (4), 365–380.
Walsham, G., 1995. Interpretive case studies in IS research: nature and method. Eur. J. Inform. Syst. 4 (2), 74–81. doi:10.1057/ejis.1995.9.
Wasserman, A.I., 2016. Low ceremony processes for short lifecycle projects. In: Managing Software Process Evolution. Springer, pp. 1–13.
Wei, J., 2017. State of the hardware incubators and accelerators in the united states [society news]. IEEE Consum. Electron. Mag. 6 (1), 22–23.
Yau, A., Murphy, C., 2013. Is a rigorous agile methodology the best development strategy for small scale tech startups?.
Yin, R.K., 2003. Case Study Research: Design and Methods. Sage Publications, Inc 5, 11.

Vebjorn Berg holds a M.Sc. in computer science at NTNU: Norwegian University of Science and Technology. He is currently working at consultancy firm Netcompany.

Jorgen Birkeland holds a M.Sc. in computer science at NTNU: Norwegian University of Science and Technology. He is currently working at consultancy firm Holte Consulting.

Anh Nguyen-Duc is a Associate Professor at the Department of Business and IT, University of Southeast Norway. His research interests include Empirical Software Engineering, Data Mining, Software Startups Research and Cybersecurity.

Ilias O. Pappas is an Associate Professor of Information Systems at the Department of Information Systems, University of Agder (UiA), Norway. His teaching and research activities focus on the areas of digital transformation, social innovation and social change, as well as Internet marketing and information technology adoption. He has worked on EU-funded projects that support SMEs to innovate, network and grow by promoting innovation through collaboration platforms. Pappas has been a Guest Editor for the journals Information & Management, Technological Forecasting and Social Change, and Information Systems and e-Business Management. He has published over 70 articles in peer reviewed journals and conferences including Journal of Business Research, European Journal of Marketing, Computers in Human Behavior, Information & Management, Psychology & Marketing. He serves as the secretary of the IFIP Working Group 6.11: Communication Aspects of the E-World. Pappas is a recipient of ERCIM and Marie Skłodowska-Curie fellowships.

Letizia Jaccheri (Ph.D. from Politecnico di Torino, Italy) is Professor at the Department of Computer Science of the Norwegian University of Science and Technology. Jaccheri's research is on: software engineering; entertainment computing; computational creativity; ICT-enabled social innovation. Jaccheri is the Norwegian representative and Vice President of IFIP TC14 on Entertainment Computing. She has published more than 200 papers in International conferences and journals. She has been teaching courses in software engineering at various levels since 1994. She has supervised PhD students, Post-doctoral students and acted as opponent for national and international defences. From 2015 to April 2018 she was independent director of Reply S.p.A, an IT company with 6000 employees world wide. She has been general chair of IFIP ICEC 2015, co-chair of ACM IDC 2018, and Program Chair of the European Computer Science Summit 2018. She participates to several Horizon 2020 projects. Letizia Jaccheri is passionate about dissemination of computer science and research to the general public and to contribute to recruit female students to computer science and research. Jaccheri is Senior ACM Member since 2017 and ACM Distinguished speaker since 2018.