CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Accelerating Software Delivery in the context of Requirements Analysis and Breakdown for DevOps: A multiple-case study

Bachelor of Science Thesis in Software Engineering and Management

Fahd Debbiche
Markus Wrang
Kundananji Sinkala

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Supervisor: Lucy Lwakatare
Examiner: Richard Berntsson Svensson

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Accelerating Software Delivery in the context of Requirements Analysis and Breakdown for DevOps: A multiple-case study

Kundananji Sinkala*, Fahd Debbiche*and Markus Wrang*

Department of Computer Science and Engineering, Gothenburg University
Box 100, SE-405 30 Gothenburg, SWEDEN
{ gussinku, gusdebfa, gusmarwr }@student.gu.se

***Abstract*—Context: Requirements analysis and breakdown in agile context has numerous challenges that is yet to be addressed. The incorporation of DevOps in agile settings inherited agile requirements engineering related challenges. Thus, it is paramount to explore what strategies are used in the requirements engineering process in agile-DevOps environment**

Objectives: We aim to fulfill the research gap by eliciting the current strategy applied by DevOps practitioner during requirements analysis and breakdown phase in industrial context, and explore the effect they have in practice. Ultimately, suggesting useful strategy to mitigate the impact or limitation imposed by current strategy for the purpose of accelerating software delivery.

Methods: An exploratory multiple case study design using qualitative research approach was used across multiple case companies operating in agile-DevOps environment. Data was collected mainly through semi structured interviews with DevOps practitioners from five industrial cases. Results: Using thematic analysis, four different strategies were identified. The effects of each of the strategies are assessed with respect to DevOps principles. From a cross-analysis of 5 company cases, five effects were identified. Negative effects of those strategies were determined in five different areas which are: delivery speed, architectural design, shared vision, requirements process, and testing.

Conclusion: The proposed alternatives are suggested specifically to accelerate software delivery in a DevOps environment.

Keywords: Agile, Continuous software engineering, DevOps, Requirements Analysis, Requirement Breakdown.

I. Introduction

A multitude of factors represented in the market with ever changing requirements and business competitiveness have caused a boost in growth of agile methodologies. Agile methodologies strive for continuous customer involvement and adaptability to change. Software organizations have allocated significant resources to set up an agile infrastructure and adapt to an agile way of working for evolutionary software delivery. Transition from traditional to agile approach was one of the means of delivering quality software at a higher pace [41].

Acquiring agile focused-perspective of software development allowed some organizations to observe the need and benefits of frequent realization of certain critical activities, such as 'release often', based on the fact that this concept is well established in open source projects [11]. Organizations have focused on addressing several barriers in order to establish a continuous engineering environment. This complex construct requires an initiative of implementing continuous integration and delivery processes. The initiative is represented in a set of actions, such as increasing the number of automated testing in a short period of time [31]. Furthermore, companies realized the need to interlink continuous integration with continuous deployment [11] in order to shorten the feedback loop with end-users. Evidently, adopting a continuous deployment practice requires multiple organizational units to be involved in this activity [31]. Nonetheless, what is important is to incorporate mechanisms of gathering constructive customer feedback and translate it to functional and non-functional requirements.

The progress to continuous software engineering addressed challenges that agile held, in particular, with the customer collaboration models. However, some challenges were persistent, mainly in requirements engineering. During this process, breaking down a requirement in order to test it separately is still regarded as a challenging task [40]. Organizations recognized, to truly implement continuous practices, an automation infrastructure is needed [40]. DevOps comes into play and promotes a tighter interaction between development and deployment. DevOps provides configuration management and automation infrastructure to compensate on delays in software releases to customer. The infrastructure builds a homogeneous environment for development and production, and capture feedback through customer's logs, which explains the adoption of DevOps especially in cloud web application and data-centric systems [6].

DevOps builds upon the agile approach and supports continuous practices paradigms, with an emphasis on interlinking different organizational silos. Mainly, DevOps fosters effective communication between development and operations as an attempt to improve software release cycle times. Therefore, it is essential in a DevOps environment, for developers to understand the real-world production environment where operational teams release code. In a similar way, operational teams need to collaborate with the developers and configure production-like development environments [40]. Indeed, operations in DevOps settings have the responsibility to handle the system configuration management, including setting up the infrastructure and pipelines required for the build, test and deployment automation. As well, operations report on the system quality aspects, such as security and performance based on the data provided in production unlike in agile development practices.

For DevOps practitioners, this represents a source of hidden product and operational requirements, as requirements are also shaped by operations. This inclusion of DevOps practitioners in shaping requirements makes resources more elastic and reduce operational challenges arising from frequent deployments. Gathering product performance-related data enable the infrastructure and operational teams to set product operational requirement [2].

DevOps caters well to continuous software engineering and agile. However, DevOps inherited multiple challenges from agile that diminishes the DevOps performance of building software. Such challenges pertain, for instance, to some practices in the requirements analysis and breakdown process. In agile settings, product owners have ownership and control the product backlog, they are the guide for acceptance, decide what to build and in what order [44]. Product owners (PO) receive requirements from different stakeholders, thus, a product backlog may handle all requirements related information. But is has been reported that product owner may ignore the recommended guidelines for managing the product backlog thus affecting product time-to-market [44]. As identified by Bass [1], six main roles of the product owner are: groom, prioritizers, release masters, technical architects, governors and communicators. DevOps brings more automated release process and equal distribution among the roles thus limiting the hand-offs that occur particularly in large software development organizations, thereby, providing feature teams with full responsibility of software development and release [24], [16].

When it comes to requirement analysis and breakdown, agile methodologies to some extent fail to achieve release cadence desired by many organizations, part of the reason is due to a bottleneck in the collaboration between development and operations which is responsible for software maintenance, delivery, and release [16]. Furthermore, frequent releases help address issues associated with continuous deployment [23]. Developing software in smaller batches makes each deployment less risky, help teams address bugs faster and detect from which deployment the error was caused. However, keeping up with frequent releases is a non-trivial task since it reflects on the requirement breakdown efficiency to keep up with continuous integration (CI) and testing [7]. Additionally, recent studies show that depicting an abstract requirement given by the customer as a deployable unit is a challenging task for DevOps practitioners as it may not fit DevOps continuous integration and deployment pipelines' frequency flow. Similarly, DevOps practitioner's target integration frequency is limited by the capability of breaking down software requirements [27].

There are seldom studies that differentiate the requirements analysis process in DevOps from agile. Further, we noticed that operation-monitoring metrics should be automatically traced back to requirements. This way, development teams and POs are able to monitor the status of the requirements in the production environment and manage them at a lower cost. In this research, We looked at different cases that adopted DevOps in a lean agile development environment and we explored requirements analysis and breakdown approaches of functional and operational requirements.

A. Research purpose

In order to provide the software community with insight into the requirement analysis and breakdown process within DevOps settings, this thesis aims to identify the strategies used by DevOps practitioners in requirement breakdown. Those strategies encapsulate a set practices and tools used during requirement breakdown process. We proceeded with a selection of relevant aspects and dimensions from existing literature and refer to DevOps practitioners' reflections to analyze the effects of the strategies used. Eventually, we seek to put in place useful strategies identified from our analysis that would help to increase the organization capability to deliver software at a more rapid pace.

B. Paper Structure

The remainder of this research is organized as follows: Section II provides an overview of the state-of-the-art and of-practices of requirements engineering practices, particularly, requirements analysis in agile development. Additionally, we introduce DevOps principles and its objectives, as well as the different approaches used for requirement analysis and breakdown in agile. Section III describes the research method applied to achieve the objectives of this thesis, presents our research questions and potential validity threats. Section IV introduces an overview of studied cases. Section V encapsulates the results of the data collection followed by a discussion in Section VI. The paper concludes with a summary and viewpoint on future research in Section VII.

II. Literature Review

The literature of focus in this thesis relates to three main concepts which are agile, DevOps, and the requirement analysis and decomposition challenges in agile and DevOps. The first sub-section introduces DevOps and highlights how DevOps builds upon agile methodologies. The second sub-section focuses on the requirements analysis and breakdown aspect in agile by presenting its status quo. The purpose of the agile requirements engineering (RE) process overview is to introduce the practices and strategies that DevOps is building upon. The last sub-section gives the state of the art of the challenges that requirement breakdown brings to a continuous software engineering environment and its effects, thus, similar challenges may arise in a DevOps environment.

A. Background

1) Agile to DevOps: DevOps uses agile as a foundation and enables continuous software engineering [26]. Conceptually, DevOps, similar to agile emphasizes collaboration between stakeholders of software development especially developers and operations. DevOps extends the agile practice of CI by incorporating deployment automation through infrastructure-as-code concept. Thus, this concept embodies configuration management tools and themes, such as automation pipelines which is a major practice used to mainstream CI, testing,

deployment, and quality monitoring [25]. Consequently, DevOps brought useful solutions to orchestrate the deployment and release processes to deliver with a short cycle time. The solutions came in the form of a set of tools and best practices with the aim to favor better agility in delivery, architecture, integration, and testing [24]. Many agile software organizations have started to integrate DevOps in an attempt to benefit from its advantages, covered by [9], and align it with the organization's high-level business objectives [37].

Gupta et al. [13] explored different DevOps attributes and assess their maturity level within a company. DevInOps was one of the listed attributes. Its domain encapsulates the communication of requirements and design from the development team to operations teams to sustain high deployment pace with improved quality. DevInOps supposedly encourages collaboration between divisions which helps correspondence and representative welfare. Continuous releases empower a more experimental approach and rapid feedback gathering. The difficulties emerging from communication structures prevent cross-department collaboration and tend to address the cultural shift [37].

Those attributes outline the benefits of DevOps in delivery and release frequency, improved test automation, and better communication. In this context, Humble and Farley [17] stated that frequent software delivery and deployment required a connection between different activities of software development. Ample evidence shows there is a need to continuously deploy software changes and gather data in the operational environment. The data is related to system performance and feature usage gathered from monitoring tools and traced back to requirements. Paulin [33] supports these claims and highlighted that, indeed, benefits were found in system quality and in release cycle time. Yet, challenges were raised regarding a lack of common understanding of DevOps concept which requires highly skilled practitioners [37].

Various literature provided insight into how to implement DevOps [32], along with its variations. Namely, on an architectural level, driven by the need to increase software delivery. DevOps addressed a key architectural barrier and promoted the adoption of micro-service architecture that supports deployment independency by a modularized development of smaller units, which in return reduced the deployment time, and enables quality monitoring through incremental quality change [5].

2) Requirements Analysis and Breakdown Practices in Agile: Requirements are a central concern in software engineering research. This concept was well studied in previous work as agile researchers have investigated different RE process challenges and best practices [19].

To be open and adaptive to changes, agile practitioners adopted several RE practices to overcome challenges emerged from the traditional way of development. Inayat et al. [18] have conducted a literature review on agile RE practices and challenges and identified 17 practices related to RE in agile methods, such as user stories, iterative requirements, and pairing for requirements analysis. The last practice encourages a stakeholder to play several roles to fill the communication gap.

Agile has treated the RE differently compared to when traditional 'waterfall' software development approach was used. Agile RE process encompasses methods and tools to handle requirements change such as product backlog refinement and impact analysis between requirements. Various agile-inspired frameworks, such as scrum employ collaborative and iterative software development with the customer. The use of different methods and supported tools of agile make software development process responsive to changing requirements and/or customer needs. Disregarding the size of the agile organization, the system functional requirement is the most commonly documented type of requirement, using free-form textual domain/business process models, free-form textual structured requirements lists or even use case models as text with constraints.

There are still challenges in the RE process in agile [43]. For instance, following user stories format in requirements documentation can be insufficient if the requirements are communicated to off-site stakeholders [15]. Bjarnason et al. [3] have identified five variants for test cases usage for requirement which poses both benefits and challenges. Bjarnason et al [3] relate to aligning goals and perspectives of requirements among different stakeholders by favouring cross-functional communication between roles, and the need for constant customer involvement respectively in terms of requirements elicitation and validation. Further, the effects of the mentioned practices in agile RE process were also studied in [20], where constant challenge regarding breaking down knowledge about customer value is reported by the researchers.

In literature there exist different approaches to requirement analysis and break-down. Racheva et. al [35] suggest a conceptual approach to re-prioritize requirements in agile. In order to produce a prioritized list with requirements, this approach takes as an input a list of relevant inputs such as work-breakdown structure, business value per feature, effort estimation, project constraints and knowledge information experience. Determining the value of a feature is a non-trivial task since requirements experts lack explicit measurements techniques that calculate the value of a requirement or a feature which affect in return the features or requirements dependency mapping.

Having a vertical of a multi-hierarchical structure of requirements are considered to have both benefits and challenges with respect to easing requirement traceability and effect on development speed. With that being said, Gorschek et al. [12] studied different requirements abstraction levels through the RAM (Requirements Abstraction Level) which encapsulates a product level, a feature level, a function level, and a component level. Prior the requirement analysis, a single requirement must be categorized in one of those levels, must relate to one of the product goals and then broken down at least to a function level and not necessarily to a component level due to the fact that having multiple iteration on a requirement leads to problems in e.g. comparing requirements.

B. Related Work

1) Impacts of Requirement Analysis and Breakdown on agile : The link between requirements and its effect on different steps of the development has been studied [42] and [22]. But then, there is no explicit research on how the requirement breakdown effect Agile-DevOps in practice to the best of our knowledge. Often requirements management represents a hurdle as requirements research falls short in identifying the scalability of the RE process [36]. Agren et al [46] explored from a managerial perspective, the requirements impact on the development speed. They found that in automotive systems, requirements are regulated by safety and legal features, thus, it implies the traceability of development and verification results to requirements. However, these aspects affects the way requirements are decomposed.

Decomposing requirements into several hierarchies is a key challenge for agility according to Agren et al [46]. Having several levels of abstraction affects the requirement change by the inability to ensure that decisions are made on the most appropriate abstraction, and make the information flow hard to trace.

In large scale software development, such as in the automotive industry context, Eliasson et al [8] reported on four different levels in the required structure. First, the functional requirements are broken down by a function realizer and expressed in use cases or scenarios describing the interaction of a customer with the car. However, some of the functional requirements providers may not have the required implementation knowledge, hence, the requirement can end-up being under-specified and the scenarios do not cover the different behaviors of the system [8]. This is mainly caused by a challenge in identifying requirements at a specific abstraction level [8].

Further literature identifies in large scale organization requirement over-scoping as one of the roots of communication gaps. This can affect requirements coverage specification, subsequently problems when implementing and verifying them arise [4]. In a similar vein, Berntsson et al [44], incomplete information is often the root cause for mismanaged items in backlog which are the basis for decision making process. Such situation hinders the collaboration of team members of doing their jobs thus affect the development speed.

In addition to that, Regnell et al [36] state that requirements complexity is proportionate to its size. Therefore, a requirement complexity can be categorized in four levels of magnitude to judge on its complexity: Small-Scale RE (10), Medium-Scale RE (100), Large-Scale RE (1000), and Very Large-Scale RE (10000). Requirements inter-dependencies heavily affect its level of complexity. In large scale RE, similarly for a very large scale, one way to depict requirements inter-dependencies is to assign requirements to different partitions and illustrate dependencies between partitions instead. However, in a very large scale requirements engineering situations further research needed to investigate how requirements are to be structured and designed, and which attributes, links and concepts need to be maintained. Similarly, the gap is yet to be filled regarding the required level of details and abstraction related to each defined requirement.

According to Inayat et al, agile RE depicts the “agile way of planning and managing requirements [18]. Only little is written about the requirements process in DevOps context. DevOps focus on bridging the gap between the development and operational teams. This involves establishing a solid requirement communication throughout the project which is a key factor for delivering what the customer wants [4]. Development teams in DevOps take into accounts various resources such as standards, organizational process descriptions and more to elicit product and operations process requirements to facilitate knowledge sharing between developer and operators divisions [9].

Explicitly, Bass et al. [2] proposed several of sources that development teams could refer to determine operational requirement, and highlighted the fact that issues related to incomplete requirements in agile projects may lead to project failure [43]. Furthermore, Bass et al argue that the development staff should provide features of use to the delivery process as well as to the operations process, since, including those features will simplify the release process and reduce the number of associated errors [2].

Yasar [45] points out to the applicability of DevOps practices in case of new requirements late in the software development life cycle (SDLC) in highly regulated environments. Normally accepting a new requirement in a later stage of the SDLC entails problems on project consuming significant time for design, development, or testing. Therefore, in order to mitigate its impact, Yasar sets restrictions on the identity of the requirement requestor. The original requestor can create new requirements based on its domain knowledge, for essential functions to the original project, and ads non-essential features to a project backlog. Upon approving requirement by Yasar [45], time is allocated for building and testing. Ideally, to handle late requirements Yasar [45] suggests to incorporate small task at a time and conform to DevOps concept of progressing the project through the entire SDLC process one small task at a time, as it contains a small number of sources to be deployed and tested.

In particular, literature reports for instance, on how continuous integration requires a link between development and operations [10]. The work reported in [40] revealed that balancing between breaking down a requirement to a small size with customer value and continuous integration is a challenging task. Similarly, breaking down a requirement to an adequate size in order to be tested separately is also a barrier. Consequently, this entails considerable delays in continuous integration.

We deduce that since these effects are persistent in agile, they are likely to impede DevOps practices. Thus, since DevOps has not been adequately studied in current literature [9], and the body of knowledge about the RE, in general, is limited [43], this study, explore in a problem driven manner, the current state of the art in of requirements analysis and breakdown strategies.

III. Methodology

This study is an exploratory multiple-case investigation of requirement analysis and breakdown in DevOps within industrial contexts. We applied a qualitative research approach to enable us to observe the phenomena in real-world setting of multiple cases, thereby increasing the robustness of the result [29]. The data was collected through semi-structured interviews and questionnaires from DevOps practitioners in five different organizations and was analyzed using thematic coding.

A. Research Questions

Literature has not explicitly explored the requirement breakdown and analysis in agile development context [43]. Hence three research questions were raised to fill in the gap of this study:

RQ1: *What strategies adopted by DevOps practitioners for requirements analysis and breakdown in practice?*

The question, based on empirical data gathered from multiple case investigation, aims to look at requirements analysis and breakdown process by exploring different activities, best-practices, tools, and roles involved in the process. Hence, this helps gain a deeper insight into requirement analysis and breakdown process within industrial settings.

RQ2:*What effects do the current practice strategies present for DevOps practitioners?*

based on the strategies identified in RQ1, this question aims to identify the effects of the overall requirements analysis and breakdown process on DevOps goals and principles .

RQ3: *What alternative strategy could be used to overcome the reported challenges and accelerate software delivery?*

This question Provides alternatives to mitigate the highlighted negative effects for DevOps practitioners to bring software to the market at a more rapid pace.

B. Study Design

We report on a multiple-case study design for this research. The case and the basis of this study, is DevOps environments within organizations. We found the companies of interest based on their DevOps environment and requirements analysis related work. Across all the cases, a DevOps environment represents the characteristics of a department or an entire organization that employ DevOps approach, including practices such as Dev-Ops collaboration, automation in deployment pipelines. The unit of analysis are companies and department, with each characterized by the latter described DevOps environment/context.

For this study, we conducted four semi-structured interviews and two group interviews across five cases, and one questionnaires for one individual case. We followed the guidelines by Runeson et al [38] to develop the interview guide. The interview guide was divided into four sections. The first section provides information about the background of the participants, the system under development. The second sections questions relate strategies of the requirement breakdown process by DevOps practitioners and the third section, about benefits and challenges of requirements breakdown on DevOps practices and fourth section, alternative strategies supporting requirements break down and analysis to accelerate software delivery. At the minimum we secured one participant per case, thus, the main core of triangulation is achieved by cross analysis of the cases [38].

Two pilot interviews were conducted prior to data collection. The purpose of the pilot interviews was to assess the validity of the guide, assure that the interview has a good structure that allows the interview to follow the questions adequately. The participants of pilot interviews possessed a minimum of four years of experience of work in DevOps environment which makes them potential interview candidate. The participants interviewed in the pilot interviews were not involved later in the study.

The interview updates realized:

1) The relevance of the questions: since the questions were categorized according to the research questions, we raised the level of abstraction of some of the questions, in particular in the beginning of each. This is done to help keep the conversation in a natural flow, with less repetition or jumps from one topic into another, and make the interviewee more talkative.

2) The clarity of the questions: some questions were further broken down into more concise questions and follow-up questions, for the aim to reduce the complexity.

C. Subject Selection

For each case, we looked for practitioners of DevOps within companies that adopts the principles and practices of DevOps. The participants' selection was based on convenient sampling. The authors referred to the company suggestions to select relevant participants to this research. The roles range from Product owners, business analyst requirement engineers, developers, architects, testers, etc. At first, we contacted a specific person to be our coordinator at each case that was working within DevOps environment, through whom we communicated our research objectives and interview guide, and how having different perspectives are useful for us. Eventually, the candidates were selected based on their preference, availability, and knowledge about the topic area. Table I gives an overview of our respondents, domain, roles, years of experience and organization team size. In order to preserve the anonymity of the candidates, we provided unique IDs for each participant.

D. Data Collection

Given the relative newness of the concept of DevOps in software engineering, the researchers aimed at gathering data from managerial, architect and developer perspective. The data was collected through semi structured interviews, semi

Case	Unique ID	Role	DevOps Experience	Domain	Departmental Size (No. of Teams)
Case P	CP-1	Product Owner	5	Platform & Tools	22
	CP-2	Product Owner	5	Full-scale Web Systems	
Case X	CX-1	Cloud Engineer	5-6	Embedded Software & Cloud Adoption	100
Case Q	CQ-1	Programme Coordinator	2-3	Embedded System, Hardware & Software	60
	CQ-2	Technical Coordinator	10		
	CQ-3	System Manager	10		
Case Y	CY-1	CI/CD Consultant	4	Software Development & Cloud Adoption	55
Case Z	CZ-1	CI/CD DevOps Analyst	4	CI/CD Development & Support	40
	CZ-2	UX Designer	3	UX/UI	

TABLE I: Demographics of participants

structured group interviews and questionnaire. One unique interview protocol was used for individual interviews, group interviews and for the questionnaire. This interview protocol in (Appendix A) was also used across five cases. However, during both individual and group interviews, the interviewers did not strictly follow the exact ordering of the questions.

1) Interviews: As pointed out by Runeson et al. [38] interviews are considered an important and common technique for collecting data. Each interview lasted one hour and was conducted by at least two researchers. With the permission of the interview participant, interviews were recorded and later transcribed for analysis. During the interviews the researchers followed an interview guide, see appendix A. The participants were invited through the company's email with the proposed time and date for the interview. For the participant located in the Gothenburg vicinity, we scheduled on-site meetings which resulted in face-to-face interviews. However, for participants in remote location such as Stockholm and Netherlands, thus, we conducted Skype interviews.

2) Group interviews: For each group interview the number was limited to two participants. The second participant was suggested by the original candidate that was supposed to have an individual interview. Initially, we aimed to limit the group interview to three people since focus groups of a large number are difficult to control [34], and also we aimed at getting each participant to answer all the interview question.

3) Questionnaire: A questionnaire was used with participant from company Y. As stated above, the participant answered the questions provided in the interview protocol (Appendix A). The researchers resort to questionnaire as a substitute for interviews due to the participant unavailability.

E. Data analysis

The semi-structured interviews provided a large amount of qualitative data. Hence, a thematic analysis approach was used to support our qualitative research. This approach is used to find patterns and themes related to our main research questions and organize the rich data provided from our interviews [39].

A total of seven transcripts resulting from our data collection with each containing at a minimum fourteen pages were analyzed. The coding process including the data visualization and categorization was done using Nvivo (see Appendix B).

Prior to coding, we determined according to our research questions, high level themes (strategies, effects and challenges, and alternatives). The word strategy is generic in a sense it is likely to be interpreted in many ways. Therefore, we defined a strategy based on a list of specific characteristics. Those characteristics were specified as conceptual themes such as practices, tools, roles and responsibilities. Later on, from the data set, we generated extensive codes. We grouped each of these discrete codes into conceptual sub-themes under the high level themes mentioned above. This is done by highlighting questions and related answers. This step resulted in a list of sub-themes and codes that were later categorized. The remainder of all the transcripts were coded in the same way. It is important to note that one code can pertain to more than one sub-theme. Eventually, after all transcripts are coded, each of the sub-themes were refined. This refinement included the addition and deduction of some codes and sub-themes for the purpose to avoid misinterpretation of the results. The Appendix C gives an overview of the initial themes and codes used in the analysis. The tone of the conversation and all informal nature of the interview was maintained during transcription and coding process.

F. Limitations & Validity Threats

This section discusses different types of threats identified for this study in according with Runeson et. al [38].

1) Construct validity: Construct validity reflects on whether the methodology and measurements adopted by the researchers

were sufficient to answer the suggested research question. For instance, the use of non-suitable data elicitation technique can distort the findings [21] making it unusable or unreliable.

The first interview participant for each case were selected based on a convenient sampling. This may introduce a sampling bias. Based on the nature of our study, the sample of the interviewees does not represent the entire population. To compensate on that, we aimed to study different perspectives form different case companies.

The context of our research requires interpretations from different perspectives. However, the number of subjects per case is limited. In other words, saturation per each case was not achieved. Instead of interviewing multiple roles in one single case, we alternatively explored different contexts and cases to help achieve triangulation. In a similar vein, the triangulation is still affected by making interviews as the main elicitation technique.

At some points during the interview, the participants had a different conceptions regrading some of the questions or did not have a constructive or a detailed answer. So, this threat was reduced by suggesting potential areas that could reflect on sample answers.

2) Internal validity: Internal validity is concerned with the collection of the data, that no other unknown cause affects the data. If participants have biases based on how we perform our interviews, the collected data will be affected. To minimize this effect, we performed all the interviews in the same fashion. The interviews are performed at a neutral place to reduce the impact the environment has on the interview. All the interviewees got a short introduction about our research, our aim and the purpose of our study to make sure they understand the purpose of our research.

3) External validity: The requirements analysis and breakdown process is context-dependant and can vary form company to another. This study included interviewees with multiple roles and teams in different companies for the aim to increase the generalizability of the results. Therefore, companies with context similar to the case companies can benefit from the results produced by this study. To help extend this research and identify similar contexts, a detailed description of the case companies context and characteristics is provided in section four.

4) Reliability: This aspect inspects that the research is unbiased and that result isn't dependent on researchers. Qualitative data brings a lot of interpretation and there is a chance that we interpret the data incorrectly [28] resulting in a faulty and unreliable final result. To make our research reliable and repeatable, we used common and tested methods mentioned previously in the methodology section to collect and analyze our data. All our steps for data elicitation and analysis was documented in a clear and concise manner to ensure reliability and make it reproducible. Further, all members were responsible for the elicitation and analyzing of the data, reducing the risk of one persons influence the data.

IV. Overview of studied cases

This section provides an overview of the cases and their software development contexts, including information of what the team/organization is developing, DevOps usage/adoption and requirement breakdown process.

A. Case X

Case X is a cloud option team from an automotive company. The company is developing transport solutions, such as trucks and buses, combined with an extensive product-related service such as rental service to enable its customers to focus more on their core business. Within this Case, we interviewed a cloud engineer, who was previously helping feature engineering teams of connected vehicles and services to automate deployment and set-up testing frameworks. Case X helps internal teams to work with DevOps and Amazon Web Service (AWS) such as building a secure service in AWS cloud environment. Case X utilizes the core platform of the company and provide assistance to about 1000 engineers across 100 teams.

Case X's requirements come from internal features teams at connected vehicle and connected services within the organization. The requirements are usually communicated to Case X by PO of internal feature teams. As such the requirements are reported as having being prioritized and refined, which makes it easier for Case X to further analyse them and begin implementation. An example of such requirement is if an internal team wants to migrate its hosted applications from Case premise in the data center into AWS.

Case X has quite high-level epics that they call stories that are discussed over a whiteboard and about five different activities or more are generated. Stories are later written on post-it notes and are then transferred to the Kanban board to facilitate further discussion and how to work on the tasks. The PO writes formal words as epics then take the epics, discusses them and then the outcome is prioritized. This is a common practice consistent within multiple teams within Company. However, for larger applications e.g connected services consisting of 30 teams, high-level requirements require the effort of all the teams and customer because they focus on satisfying end-customer needs. Hence teams are built on specific use case or service for easy and direct customer feedback. Testability, batch size and ease of monitoring are among three criteria used by Case X to assess break down of the requirement. Weekly demos are held to demonstrate what they had made from the previous week, after that, they set a day for a session for requirement refinement.

Requirements are decomposed and detailed in JIRA or post-it notes on the wall in order to track all the activities done within the group. However, there is no unique tool within Case X used during this process as this varies from team to team. A dateline of one to two weeks maximum is set as completion of these tasks after the breakdown and implementation, the product is shipped into production. Thereafter new small action and stories are formed again based on the feedback from the end-customers and the process repeated.

B. Case Y

Case Y is an IT organization with banking licenses rather than as financial institutions with IT departments. The Company provides financial services and advice to private, corporate and retail clients through creative solutions to suit their clients business needs. A CI/CD consultant from this Case was interviewed and belonged to a central team providing a container-based platform on Amazon Elastic Container Service for 55 internal teams wanting to deploy their applications on containers. The interviewee's responsibility within this central team was requirements gathering for the implementation of new tools and processes as well as building a container platform for an entire software development teams within the organization and doing migrations of tools to the cloud.

The infrastructure team defines the requirements themselves through the PO based on what happens in the market and what could benefit the Case. The requirements are usually written as text and are based on stakeholders intuition. However, there are some requirements that are derived from problems and issues which happen on a daily basis within the organization e.g. lack of security steps (Hashed against plain passwords in git) or from slow release cycles.

Requirements are split between functional domain, such as: what the tool should do, which security aspects are needed, which enterprise-grade features are needed, what the amount of work is to maintain it and the level of automation. As such, the requirements become stories and later these stories are decomposed to task. Within the organization, a small refinement session is performed and later development begins. If issues occur, for example a fault tool, a technical session is held mostly via video call to discuss with vendors and other stakeholders. The central teams have high-level epics in Jira and they are detailed in confluence.

C. Case P

Case P is a telecommunication company that provides customized local and global telecommunication services and solutions. We interviewed two POs from Case P. The first PO is responsible for the platform and tools team and the second PO is from the sales development teams. The platform and tools team is a dedicated IT department responsible for providing DevOps tools and maintaining the DevOps platform for different teams (about 22 teams) working in a CI/CD environment. The platform and tools team consist of three sub-teams, of which two are infrastructure and tooling development teams and one is a process-oriented team. The sales development team is responsible for the whole sale process done on their website for specific products. The sales team consists of a solution architect, UX designer, testers and, front and back end developers. Both teams typically work in agile manner with team sizes between 8-12.

During the requirement gathering phase, requirements for the team come in form of small issue or a large issue e.g. building a continuous delivery pipeline on top of legacy. In contrast, for the sales development team, the requirements come from different sources such as internal business unit, marketing unit and special cases from external customers as comments/feedback. Activity of specifying requirements involves the product owner and a group of architects that rather rely on conceptual reasoning in identifying both the structure and how a single requirement should be implemented rather following a requirement specification template.

Requirement breakdown was done differently from both the teams. For an agile sales development team, the Product Owner is only involved in customer requirement level and as for technical task level, the solution architect breaks it down and works with the developers to implement a solution. In contrast, platform and tools team used to be a requirement driven team but now working in agile kind of way breakdown the requirements into smaller chunks starting from the epics.

The sales development teams and platform and tools POs both materialize the user stories and document them using a set of tools such as JIRA, emails, PowerPoint, MS Excel and Case P's Confluence which are later prioritized. Case P's Confluence is used to document technical related details on how things are being developed. It outputs a developers handbook and operations handbook. On the other hand, Jira is used for more formal documentation such as instructions.

D. Case Q

Case Q is a large scale multinational networking and telecommunications company. Case Q is developing software and hardware solutions for the telecommunication industry mainly 4G and 5G networks. The company has adopted a DevOps program that targets to improve the release cycles and efficiency. In case Q, we interviewed a System manager from System Department Early Phases Program, Technical coordinator and programme manager at R&D product development unit. Responsibilities include finding new strategies in improving the ways of working, wherein DevOps program is established to improve the collaboration between development and operations. Among the goals is to ensure that development teams efficiently get feedback based on customer data e.g., product performance benchmarks to further improve system functionality. This is in addition to finding ways of employing modular (microservice) architecture while building on established CI and CD practices.

The telecommunication industry is regulated in terms of software development. You can only develop something that is in the standard requirement which comes from the standardization (e.g 3GPP) c. In Case Q requirements comes in two different types. Often a big part of the requirement and feature provision is based on the standards. Those requirements are created by the product managers. The Product manager gets the requirement from such standardization source in order for Case Q to position its superiority in the market.

Case Q's requirement process is long all because of standardization and regulations. A feasibility study of one to two pages of high-level text describing the outcome of what they want to achieve is conducted. The proposal is then studied by the system team to understand what changes are needed and

what at cost. Thereafter, Product Manager decides if the study items are feasible to be implemented. A number of detailed standards documents are produced to answer and describes how things should be implemented. DevOps set up is based on pre-established CI practices and test machinery, that execute the basis of 50,000 test cases every week and Currently, the organization has the capability to delivery from mainline on weekly basis to up to 10-15 customers.

E. Case Z

Case Z is a large scale company working with different business solutions such as salary and HR systems. The Case has worked with CI for about 4-5 years and they have recently started with continuous deployment in several departments within the organization. The goal of implementing DevOps is to enable frequent deployments and later increase and improve customer feedback in their development workflow. In this Case, we interviewed a developer working in their infrastructure project that works with other companies, helping them with technical challenges like on improving the delivery process and a UI/UX-designer who, together with the PO, form and break down the requirements specifically those concerned with User Interface.

The PO handles the requirements before they get provided to Case Z's different development teams. The PO are responsible for a set of requirement that is either suitable for the development teams from preliminary work documents or made from by business people covering what the customers would want. The product owners get support from various roles in decomposition phases, such as UX designers and Case experts. Further, it also ensures that the POs requirements are accurate to Case Z guidelines.

After break down of requirements by the PO hands over to the development teams who further discuss on how to prioritize and handle the requirements via team meetings. A requirement varies a lot in size and could take from one iteration to five or six to implement. The different levels of the requirements are legendaries, epic, and features which are a part of the epics. The features are added to the backlog as one or more product backlog items and all backlog items contain tasks which the developers start to develop.

V. Results

This section encapsulates the findings of this study resulting from the thematic analysis of the data collected through our interviews. The results are presented according to the main research questions.

The first research question provides insight into industrial practices of requirements analysis and breakdown process by exploring what practices, tools, and which roles are involved in this process. The second part of the analysis answers the second research question by highlighting what effects do the strategy has on DevOps practitioners. Eventually, we suggest a list of potential improvements on requirement breakdown strategies that could enhance the software delivery pace.

*A. **RQ1:** What strategies adopted by DevOps practitioners for requirements analysis and breakdown in practice?*

The strategies illustrated in this first section are adopted by teams working in agile-DevOps environment. There is no particular way a company must follow to establish DevOps, rather adopting principles and practices to comply with DevOps capabilities. Those capabilities not only on a practical level, for example, automating continuous integration, delivery and deployment, but also on cultural level where a new mindset, behaviour, and skill set is required to cope with this new style of software development. This section will elicit the requirements analysis and breakdown process in agile-DevOps environment among the five case companies. The purpose of this, is to scrutinize changes made for altering requirements process in industrial contexts in order to adapt to DevOps.

Requirements analysis and breakdown activities take various types of requirements as an input. This process include activities that check the requirement consistency, clearness, and feasibility with the provided resources. Furthermore, the analysis process rely on a set of practices such as requirements negotiation and prioritization to handle requirements conflicts. As observed in the cases, infrastructure requirements are also considered as key both in development and accelerating the delivery of software changes.

We report on four identified requirements breakdown strategies, namely:

- Hierarchical-based
- Feature-based
- Feature-based
- Value-based

The strategies identified are blended with a set of best practices and tools.

*1) **Hierarchical-Based Strategy**:* The hierarchical based strategy is observed within Case Q. This strategy is characterized by a top-down, sequential requirement analysis and breakdown. It comprises phases, during which specific roles perform specific activities on requirements. Interestingly this strategy is observed to co-exists with the data-based strategy (discussed later).

During this lifecycle, the requirement goes through a couple of iterations before it is specified and systematized. First, the requirement is described at the highest level along with its outcome. One example of a requirement is "Increasing the number of users supported by the system to a certain magnitude". The reason for still employing traditional requirements engineering is due to the constraints, such as standardization, complexity of the system, and a highly regulated industry.

Multiple cross-fuctional/domain experts called system managers provide initial specification of the requirement through a feasibility study. Afterwards, the product manager provides the initial breakdown of the requirement into deployable items. After the implementation items have been specified, system managers, who are more knowledgeable about the architecture perspective of the overall system, produce detailed documentation of how the requirement should be implemented.

Strategies	Practices	Roles Involved	Case
Hierarchical-Based	• Align Test and Requirement • Continuous Customer Involvement • Document Update • Feature Analysis • Requirement Specification • Requirement Management	• System Architect • Product Managers • Developers	Case Q
Feature-Based	• Continuous Customer Involvement • Feedback Driven • Requirement Refinement • Requirement Specification • Cross-Functional Team	• Cross-Functional Teams • Product owner • Developers	Case X
Value-Based	• Document Update • Feature Analysis • Requirement Management	• Architect • Product Owner • Developers • Business Analyst	Case P & Z & Y
Data-Based	• Align Test and Requirement • Continuous Customer Involvement • Document Update • Feature Analysis • Requirement specification • Requirement Management • Cross-Functional team • Feedback Driven • Requirement Refinement	• Cross-Functional Teams • Product Owner • System Architect • Product Managers • Developers	Case Q & X

TABLE II: The table shows a list of identified strategies within the case companies. The strategies encapsulates a set of best practices

> "*Requirements are analyzed by the product analyst and the product owner, basically the people owning the budget. They then consult with the system department with those requirements and then the iterative process starts from there where we have dialogue with the end users so there is not so much, that aspect is not so digitized*" (System manager, CQ-3)

The system managers get in this process and study the cost and effect related to this implementation. At a second phase, based on the inputs provided by the system managers, product managers decide upon the feature implementation.

> "*The product managers, in the beginning they do not go to that level, they put it a bit on the abstract level, and then we have what we call system managers who are more knowledgeable about the product details who then go deeper into these high level requirements and break them down into implementation items.*" (Programme Coordinator, CQ-1)

The deployable items description is specified in details and documented to ensure that the developers follow the implementation plan accordingly. After the first decomposition, the requirements are read again by the system managers who speculate on the implementation logic. Our interviewee reported on an extensive level of details provided in the requirements specification and systematization. The requirements specification are documented in specific detailed documents which is provided as inputs to the development teams. Thus, this applies a traditional process of requirement specification during which the analysis and decomposition of the requirements are in charge of a limited amount of stakeholders. Project managers and technical coordinators have to come up with a consensus on what needs to be implemented and how.

> "*Once that done we need deeper study on how we want to implement this, and then there is a number standards documents that we produce which answer and they are very detailed and describing how things should be implemented*" (Programme Coordinator, CQ-1)

The documents produced in the requirements specification and systematization highlights the requirements dependencies at an early stage. System managers highlights possible requirements dependencies by depicting higher level and lower level requirements. Before it is handed into a specific department(multiple teams).

"*When you have one kind of a program view and the program has a certain purpose, certain phase and a set of functional responsibility. But then it could become a cross program requirement. Then different stakeholder can have different priorities. Then it becomes like a negotiation, can I borrow a thing from you?*" (Programme Coordinator, CQ-3)

Departments in Case Q works for either a long term strategy or a short term strategy. The feasibility study output which specifies the spectrum and the implementation of the requirement. The requirement aggregation take into account which teams are available and at which date the requirement is planned to be delivered. The requirements are later broken down to a functional level in accordance with the system architecture and handed it over to the development teams. This also highlights the systematization effort estimate. System managers coordinate with the cross-functional teams. In a series of meetings, the requirements are then reviewed for possible conflicts and refined and implementation is aligned with the system architecture. On a low level, our interviewee highlighted the implementation details get addressed in solution agreement meetings. The meetings involved the developers and their direct requirements authors, hence, more dynamic teams which include roles varying from developers to product owners to architects. We noticed that no formal documentation produced to trace requirements change. Instead, most of the process is handled by communication.

" *We usually start with describing the requirements and working it into a business case and also what kind of extra things that we need. And then we dig further into the systematization work and then a certain point have those texts that can vary. we talk about sprints of 6-7 week of time if you want to do systematization. I can say that the best setup you can have is that you are aware about where the cross functional teams are able to receive a new requirement*" (System manager, CQ-3)

In summary, the analysis and breakdown process heavily relies on the product managers and system departments to whom falls the reasoning about the requirement implementation in the early development phases.

We report on other constraints that reflect on the decomposition process. This highlights the decomposition process takes into consideration resources, such as the budget and number of teams.

"*Product line have a certain frame, a certain budget and that, correspond to certain amount of teams. It's about what the team should be fed the next sprint.* " (System manager, CQ-3)

Weekly solution agreement meetings are conducted between development teams and system people to discuss further the requirement specifications and interdependencies.

"*If still are any gaps than this gets addressed in the end. It's when like discussed in implementation because it is a hard hand-over but they do what we call solution agreement in meeting*" (Technical Coordinator, CQ-2)

This entire process limits in a way the role of the developers and technical teams. The developer does not play a role within the process apart from reasoning about low-level requirement details to come up with implementation decision of a task. Further, we noticed that not all of the information is provided to the development teams.

"*There's just a little discussion I would say during the handover between the system managers and the developers*" (System manager, CQ-3)

Consequently, based on the complexity of the system and sub-systems the developer is bounded to specific tasks, thus, developer task scope is very limited.

"*If you are a developer you get a requirement and you need to have a implement these requirement in the code make sure that you pass all the testing all the verification needed then your job is basically more or less completed form that and then, the developer loose the oversight of what they have done and internally behaves in the feed this is one aspect*" (Technical Coordinator, CQ-2)

JIRA is the tool for task management as well as other internal tools developed internally by the company.

2) ***Feature Based Strategy****:* In feature-based strategy, the requirement analysis and breakdown output focuses on creating features with the ultimate goal of examining how the different features behave and used. Development is relatively fast and incremental in comparison to above Hierarchical-based. Assumptions and speculations on requirements are observed in this strategy. This strategy was observed for example in case X and Z.

"*We have product specialists and we have sales-people. And then together with the product owner, they usually decide or I have a feeling for what the customers want.*" (CI/CD DevOps Analysts, CZ-1)

For Case Y they start with an idea, this idea gets analyzed and requirements get set. They send a request for proposal and create a proof of concept. They then install the application in the development environment. The requirements get discussed, tested and accepted and are then ready to go into production release when all the requirements are met.

The requirements have different abstractions that at higher level constitute features, epics and legendaries. Figure 1 gives an overview of the different abstraction levels.

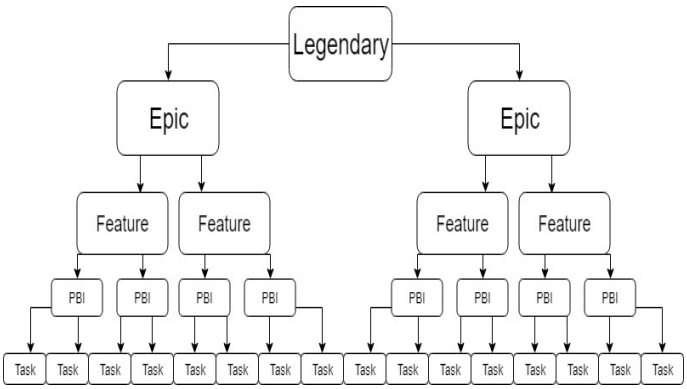

Fig. 1. Requirement abstraction levels

The requirements start as legendaries or epics which are decomposed into smaller tasks as the team then can start to work on.

> "*We have quite high level epics or we call them stories. Those are then discussed in a whiteboard and then we can probably have for each story an average of five different activities more or less only written on post-its and then they come up to our Kanban board and then we start to discuss and work with those.*" (Cloud Engineer, CX-1)

The legendaries and epics are handled by the product owners. Legendaries are broken down into epics and the initial break down of the epics result in features which can be added to a product backlog. Product backlog items are later created that contain these features and is handed over to the development teams. The development teams, later, decompose the backlog items to tasks.

> "*But we try to get the team involved, at least at a feature level. Maybe even at an epic level, but it depends we have meetings where we discuss this features like what it give to the product or service and how to maybe time estimate and, discuss what kind of features we're prioritizing depending on some kind of return on investment or time management.*" (UX Designer, CZ-2)

The main requirements analysis work from the feature level is handled by the cross-functional teams. The product backlog items prioritization is under the teams' responsibilities. Team autonomy has an impact on the requirement analysis and breakdown as the team analyze and breakdown the requirements independently of any other team.

> "*We have, you could say every week we have a meeting with the team where we define and talk about the most prioritized backlog items that we need to do*" (CI/CD DevOps Analysts, CZ-1)

The requirement they set in Case Y get prioritized based on four different prioritization levels "Must have", "Should have", "Could have" and "Won't have" and if the requirements are unclear the team attempt to clear it up as quick as possible through a technical session.

> "*We create the requirements and the task or story. After a small refinement session we just start working on it. If problems arise we will tackle it right away. If needed organize a technical session with a software vendor*" (CI/CD Consultant, CY-1)

The conflicting requirements are handled by either the chief managers or the product owners. They have meetings with other product owners and chief managers to talk about what the different teams are developing and deal with potential conflicts in requirements. However, on the team's level, the scrum master identifies possible requirement dependencies.

> "*I think, it's mostly up to the product owners. They had some meetings or a structure for discussing what they are doing now and what they were focusing on, but happens that things get built twice or three times.*" (UX Designer, CZ-2)

In case Z context, the scrum master is in charge of identifying requirement dependencies and work on how to handle them.

> "*We have our scrum masters, a big part of their role is to identify dependencies and try to isolate them or not, not to have as many dependencies* " (UX Designer, CZ-2)

*3) **Value-Based Strategy**:* In the context of the analysis and breakdown of the development infrastructure requirement, we report on a value-based strategy of requirement analysis. This was observed in company P. In this case the aspect of value is different since this strategy concerns the analysis and breakdown of system operational requirements. Operational requirement are driven by security and performance aspects. Thus, a value in this strategy form a product manager perspective should be a list of task on the product backlog that must affect the way the system behave or the development teams delivery pace. One interviewee responded :

> "*What value we can provide by making the teams more efficient. We started four years ago, providing a new pipeline for the teams. So they went from like, 30 or 40 releases a year or to, 3000 releases a year.*" (Product Owner, CP-1)

Requirements are often provided by the development teams that are using the infrastructure. Therefore, they represent the original authors of the requirement. The requirements are more often specific and technically well described.

> *"It's not very abstract. It's very concrete."* (CI/CD DevOps Analysts, CZ-1)

The analysis of the gathered requirements is critical in this strategy. This is conducted by the product owner and a group of architects. The requirements are provided in the form of issues. The analysis takes as input the reported small an big issues. Afterwards, these issues are regrouped separated by the their domains, thus, each service teams will receive a predefined requirement. Each requirement constitute a set of tasks which represents the reported issues.

The product owner relies more on conceptual reasonin g in materializing tasks. When asked about how does the breakdown and requirement structure look like our interviewee quoted

"Normally I have my brain, so that's what I try to use" (CI/CD DevOps Analysts, CZ-1)

As mentioned in the interview, product owners and architects usually write requirements and user stories. It is up to the team to prioritize and reflect on the low level implementation details. The product owner initially fills the product backlog with epics, which are large grained chunks of business value, before it is being handed it to the developers. Moreover, the product owner is in charge of the product backlog update and task prioritization.

"Product backlog for me as a product owner for one of the team's where I try to write down our requirements (CI/CD DevOps Analysts, CZ-1)

The requirements refinement occurs every week. During which a scrum master of each team meets the product owner to work on requirements aspects that are not clear.

"The scrum masters that provide very much clear and direct communication to me if something does not work. So that we have the refinement process every week (CI/CD DevOps Analysts, CZ-1)

JIRA and Confluence are used for documentation and activity tracking. However, our interviewee mentioned the impede the process in a case of driving complex systems and handling requirements conflicts. The tool does not provide a requirement or epic traceability and coordination.

"We tried to document that in the JIRA, this sort of documentation, but they are very shortly documentation. If you want the instructions, you have Confluence. JIRA system is absolutely terrible. (Product Owner, CP-1)

4) ***Data-Based Strategy****:* this strategy can co-exist with the previous strategy especially it constitute to new source of requirement. In data-based strategy, the data of the system overall performance reported from customer logging is what grounds the requirements work. For instance, in Case Q, requirements are shaped by the systems response in clients network configurations. This is handled by the operational infrastructure, where customer's feedback of product usage is observed and collected. This strategy, is characterized by a specific type of data that is collected from the infrastructure. These data can be in a form of metrics that indicate the performance of a requirement. A good traceability in requirement decomposition (such as the hierarchical strategy), makes the breakdown of the newly updated requirement less risky.

" *Development should have possibility to see feedback the performance of the code commit and for the problem in all these kinds of levels and get data back to be able to analyze and improve for further development of features and functionalists that they working on*" (Programme Coordinator, CQ-1)

Performance metrics data reported from logging constitutes the basis of the systematization of the requirements. This feedback is necessary to report on the behaviour of the feature under a specific configuration which cannot be fulfilled by the tests suites.

"*And the reason for that is we need feedback on how it works in a commercial network configuration environment, in real environment*" (Programme Coordinator, CQ-1)

One aspect important for Case X is that when developing a feature it should be able to be monitored, thus they can get data on how a specific feature behaves. The data helps them when analyzing and breaking down new requirements. Data such as system behaviour helps them on what to focus on and can help them to detect inaccuracies within requirements.

"*One thing that also is very important when we build, evolve our services is that it should be able to monitor that the feature is used, that the feature is behaving as expected*" (Cloud Engineer, CX-1)

One important aspect is that they prioritize the features they implement based on feedback from the customer. The requirement update is also based on the this feedback .

"*So this important point to have a continuous discussion about what this is the most important thing to do. So this should encourage teams to prioritize based on end user feedback. So we started to do a work it happens to see that the customer didn't want it. We shouldn't then just to finish a story because we have written it, we should drop it continue with the work and continue what now looks to be most important.*" (Cloud Engineer, CX-1)

Case P gets continuous feedback on how their product is used by customers. Through the product, users can give feedback on the system. The feedback is visible to the development teams.

"*We have on the web page, we have places where the customers can, give us a score and also comment, we have a channel in Mattermost where we are receiving customer comments and the customer score of the area we are responsible for so that, that is one kind of input we get.*" (Product Owner, CP-2)

Case X follow different feature trends and focus their work on what feature is used the most by the customer. They get the data by monitoring the different features they implement so an important aspect when analyzing and working with the requirements is the monitoring part.

"*But, in general, is very much about trends but if we see that a feature is being used more and more that it is a positive trend then we encourage the team to continue explore that area.*" (Cloud Engineer, CX-1)

B. ***RQ2:****What effects do the current practiced strategies present for DevOps practitioners?*

In agile environment, the incorporation of DevOps brought new guiding principles. Thus, the list of the identified strategies is perceived to have effects that hinders DevOps and the attributes that influence its implementation. This is done with regards to papers such as [16], [30], [13].

Thus, one relevant concern of this study is the effects of the adopted strategies on DevOps attributes and principles.

In order to discern the effects of the applicability of the strategies in agile-DevOps environment, this study examines the impact of the strategies. Table III illustrates the link between identified strategies and their effect with respect to RQ2 (The + sign indicates a positive effect while the - sign indicates a negative effect). Figure 2 depicts how each of these effects affects the 5 high level themes we formulated from thematic analysis with respect to DevOps. We present the description for each high-level theme with accompanied representative quotes of this section.

Strategies	Effects
Hierarchical-Based	+ Requirement Specification + Requirement Dependencies - Autonomous teams -Lead Time
Feature-Based	- Cross System Requirements - Under specified Requirements + Lead Time - Assumption and speculation - Design decision -Shared Vision
Value-Based	- Early requirement Conflict - Requirement interdependence
Data-Based	- Decomposition - Requirement Traceability + Autonomous teams

TABLE III: The table shows a link between identified strategies and their corresponding effects

1) Delivery Speed: The results of this case showed that the current strategies in use affect the acceleration of delivery speed. The effects mostly noted were how fast and more frequently releases and deployments were done. hierarchy-strategy is characterized by a waterfall requirements process due to many bureaucratic steps from different departments. Contrarily, data and feature based strategies provide faster feedback by continuously integrating minimal task hence satisfying the customer through early and continuous delivery of valuable software. When asked about the DevOps and proper requirement breakdown and analysis set-up technical coordinator in Case Q posits the idea that they are able to execute 50,000 test cases every week and deliver upon that to 10 - 15 customers weekly.

Our interviewee confirmed that although at the beginning of breakdown process is always slow, it later lead to faster delivery since templates and best practices are re-used for other feature improvement based on the feedback from the logs after production release.

> "*Frequent release is very important and has very big impact since, the better requirements are set up, the faster and more reliable the release process will be. Once stuff is worked out and deployed to production release for the first time, things will only need maintenance and upgrades etc. If requirement analysis not done properly this takes really a lot more time than expected*" (CI/CD Consultant, CY-1)

Our interviewee reported that it happens sometimes that they have some problems with splitting the issue to the correct size.

> "*That development stopped actually because of splitting requirements that was, contradictory and they could not fuse them in the answer. Then we stopped development on that issue*" (Product Owner, CP-1)

> "*Every other week we are delivering ten continuous deployments to customers, that actually load in their commercial network in certain area. With that way we even more feedback on the quality and also the new functionality*" (Technical Coordinator, CQ-2)

Consequently, the interviewee noted that operational requirement breakdown is overlooked during breakdown analysis and thus there is slow time-to-market, increased lead time and slow releases.

> "*Most of the time overlooking while this is an important factor to consider. Remember: once stuff is worked out and deployed to PR for the first time things will only need maintenance and upgrades. If requirement analysis is not done properly this takes really a lot more time than expected.*" (CI/CD Consultant, CY-1)

2) Architectural Design: For faster integration and delivery within a system, an architectural consideration is necessary in requirements analysis and breakdown, since they have a different architecture for the subsystems and different architecture for other systems. In the case of hierarchy-strategy, requirement breakdown requires inputs from architecture prior to implementation and architecture highly influences the breakdown process. In contrast, in a feature-strategy the autonomy needs to ensure minimum conflict among each other. That there were instances where 20 people were working on one issue and it was difficult to know what each was doing and which task was completed. Therefore traceability of each task is thus eventually left to each practitioner.

> "*As the task that we are doing do not really have to be really high coupled to a specific task if for example we talk about Jira there is no requirement that something has been made should be connected to a specific task of course the stuff that has been made was the outcome of a discussion which is important to do but having that perfect traceability of requirement i would not say that something that we give so much value on*" (Cloud Engineer, CX-1)

Feature-Based strategy encapsulates architectural forums for the cross-functional teams. The teams address requirements

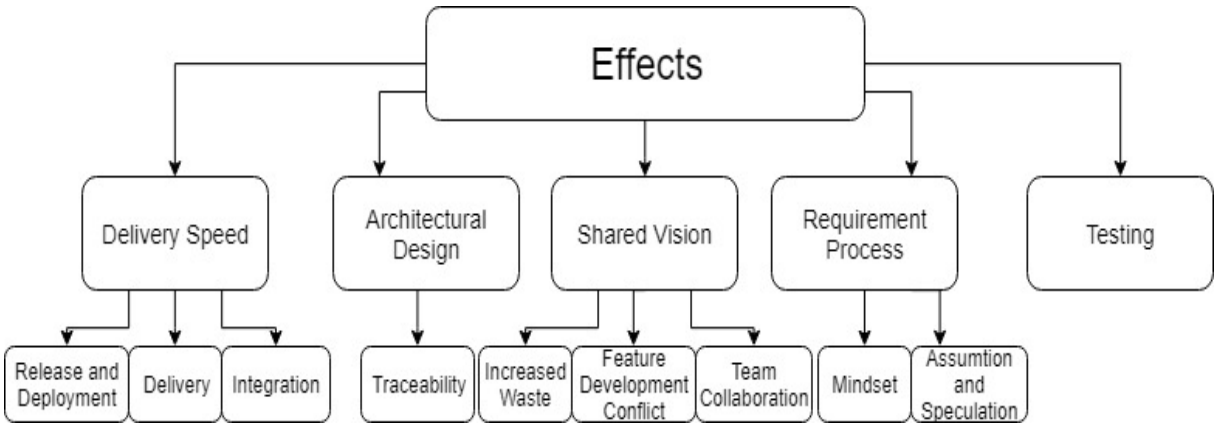

Fig. 2. Effects of Strategies

conflicts on either a team's level or higher level requirements conflicts.

> "*Absolutely on the architecture decisions are made within the teams but there are some scenarios where teams in the same domains needs to discuss all the team borders. So there are, there are a couple of architecture forums, where at least one team member from any team are gathered together*" (Cloud Engineer, CX-1)

Nonetheless, In strategy Q large complex systems are built on top on the legacy system and different levels of requirements are discussed, from the architecture perspective, nothing changes but even if requirement break was done properly done. In the end, it depends on the person that is going to implement the application which architecture to consider.

> "*The architecture has been set 10 years ago. We are not change the architecture, so there's the basic functional level. If she wants anything that is added extra, is controlled by license or feature. But the conflict comes is two features is enables in the same time, which produce contradicting result.*" (Technical Coordinator, CQ-2)

Similarly, in a data based strategy the focus on requirement breakdown is limited. Therefore, most of related issues resulting form this activity is backed up in architectural forums. Those forums are handled one to two times a week.

From the interviewee perspective, architectural design consideration should be prioritized in early design decision to avoid jumping into conclusion and miss to oversee the entire set of requirements or solution.

3) Shared Vision: Three effects were identified as the outcome of breakdown and analysis are relative to shared vision which are increased waste, team coordination and feature development conflict.

In feature-based strategy, during the break down process, different teams are asked to work on different features simultaneously. This is usually ends up with conflict, especially in cases which where e.g value-driven sales development team wants to release a product quickly.

> "*I said this is really a large organizations. Sometimes features can not go together. So let's say that a product manager wants to have a feature implemented and another product manager wants to have another feature implemented, sometimes we have features who are conflicting against each other.*" (Technical Coordinator, CQ-2)

This is evident enough that once requirements are split this way, there is a need for coordination between managers and developers in such a large organization. The requirement breakdown at a certain level does not highlight feature dependencies. In a hierarchical based strategy, the development teams does not interfere in this activity thus they loose oversight after the integration of the code. To compensate on this and minimize risks in the integration, in a hierarchical based, this challenge is addressed by the practice of feature toggles. However, this is handled in a more ad-hoc way in a feature based strategy though weekly meetings.

However, the interviewee expressed concerned on how large teams from his department had a high focus on creating autonomous team which relied on using forums as way to coordinate between teams. This lead teams reinventing the same wheel multiple times.

> "*It is a challenge to be efficient because there are a lot of the teams here in different departments, but I think it's mostly up to the product owners. They have meetings or a structure for discussing what they are doing now and what kind of, what they were focusing on, but it happens, that things get built twice or three times.* " (CI/CD DevOps Analysts, CZ-1)

In the value based strategy, each requirement formulated is analyzed then addressed to a particular team that is responsible for a specific service. The teams are more autonomous , in a sense they have more the freedom to work independently or try out new innovation. However the interviewee pointed out that it reduced communication and hindered the team share vision.

> "*When the teams are encouraged to work very autonomous could also be harsh miscommunication between the teams when they are parts that are interconnected in terms of a system that the end user faces so the miscommunication is also one thing that is challenge for us*" (Product Owner, CP-1)

In some autonomous teams, breakdown process was mostly based on assumptions where the team assumed that there's a value if feature A or B for example is implemented. However, there's no guarantee in the end based on that theoretical assumption that things will be used in reality and lead to increase waste.

4) Testing: Our respondent thought that one criterion in analysis and breakdown is that a requirement item should be testable but in large complex systems there are several levels of testing and this has to be optimized for better testing execution time. In hierarchical-based strategy the focus and effort of analysis and breakdown enables the developers and testers to execute a high number of test cases daily. On the other hand, for feature-based strategy and value based strategy aiming to perform higher regression and unit tests has put pressure on the breakdown work. Also, the developers tend to lack precision in developing accurate testing even when they are responsible for testing.

> "*So we have software main track where we have all the designers that delivering code into this and then we have all the different test level: block level module level, on complete node level, on complete RAN level so we have daily feedback from this, we tried to improve this feedback and make it even more visible and better and automatic so it can be more efficient than quicker when we have the quality in place for our main track, which we planned to have daily actually. But sometimes it's a bit difficult with all this thousands of people develop and delivering code into it*" (Programme Coordinator, CQ-1)

Despite the difficulties caused by the frequent testing, practitioners are still encouraged to commit small units of code to the main machinery from break down process as reported by the interviewee. this is observed to be implemented better in the hierarchical strategy.

> "*Its all in the last phase when we want to say the requirement is going to be implemented. So how should we add a new configuration needed or if test needed to verify new requirement into our CI loop that's only place where we get you know, this discussion*" (Technical Coordinator, CQ-2)

Only in the value based strategy, interviewee indicated that the smallest unit of the requirement breakdown must be testable.

Life cycle management issues arose a few years ago due to a lot of backlog and legacy. Managing technical debt is problematic for software testers, especially those organizations that develop and maintain large software systems however, there was always room for improvement as reported by the interviewee.

> "*But there are situations when I met teams where we see applications have a high technical debt and that is a typical situation when a team have a highly stressed requirements and requirements are pushed very hard with time estimate and those are the one that suffers from hard system that is hard to maintain and so they are slow to develop during the time.*" (Cloud Engineer, CX-1)

5) Requirement Process: Some teams have a tremendous responsibility in the testing or putting down the business requirement into the technical requirements. Our finding in this research was that requirement specific process such requirements assumptions and speculations challenges and mindset were the effects from strategies used by DevOps practitioner. If the requirements are not specified in accordance with the customer needs, it impacts the product negatively.

An interviewee said that sometimes they get used to working in a certain way, and the whole breakdown process is a challenge to them hence they have a motto "if it does not break do not change it".

> "*That process is kind of a challenge at some times because the teams are not too interested in,new ways of working*" (Product Owner, CP-1)

Consequently, a lack of interest from the practitioners of DevOps, because they deem the process as overwhelming in the case, were the requirements, in the beginning, were too abstract thus putting so much pressure on the practitioner to break down and materialize task.

> "*We sometimes misses information because everyone is so engaged and his style is completely frustrated with the task he is doing. He has no time to look little bit, on what is happening on the or the related areas to get the fairly complete picture*" (Technical Coordinator, CQ-2)

C. ***RQ3:****What alternative strategy could be used to overcome the challenges and accelerate software delivery?*

This section answers the third research question of this study, what alternative strategy could be used to overcome the negative effects discussed in by RQ2. Table IV gives an overview of the challenges linked to the strategies from RQ1. The presented results are a cross-analysis of what each DevOps practitioner outlined. Hence they are not specific to each case.

	Delivery Speed	Architectural Design	Shared Vision	Requirement Process	Testing
Hierarchical-based Strategy	X	X			
Feature-based Strategy	X		X		
Value-based Strategy	X				X
Data-based Strategy		X		X	X

TABLE IV: Mapping of challenges

Hierarchy-based strategy towards more feature-based strategy were the recommended strategies to address effects on delivery speed, shared vision and requirement process. Two respondents thought that in order to address these effects, autonomous teams should be self-sufficient, central and have enough freedom to make good choices while monitoring how teams are working, on what details and how marginal the scope is. Furthermore, it was highlighted that teams and developers should be close to the requirements discussion in order to build the correct applications that were scalable and maintainable.

> "*Give teams enough freedom to make good choices but keep an eye on teams which are working in too much details which will slow down development*" (CI/CD Consultant, CY-1)

Assumptions and speculations in requirements which is an effect from requirement process are better addressed by data-based strategy. The Respondent thought that there is need for autonomous teams not to rely on Product owner to breakdown the requirements but formulate refinement sessions, structured meetings and specific forums in order to acquire the technical knowledge to break down the requirements. This way everyone will have a good understanding of the thing that is being built.

> "*I would say the only way to accelerate this is to understand very well what is, how the products are being used in the field.*" (Technical Coordinator, CQ-2)

One aspect mentioned by our interviewees was that delivery speed could be accelerated through feedback the teams gets from the customers via integrating a build system that takes customer logs and categorize the issues that the customers have. From this, requirements are created and fed back to the backlog and at the same time removing logs that have no value through cost-benefit analysis. Feature-based strategies could address this aspect.

> "*I was coming back to the same things. Try to get back to that weeding out non valuable things from the backlogs. Then we would have a much smoother delivery process. We do a lot of non valuable stuff.*" (Product Owner, CP-1)

Data-based strategy while using prototyping was identified by the respondent as one way to remedy the outlined effects. A discussion of different implementation aspect beforehand with the focus on user experience metrics is done before even development begins. The purpose of the discussion is to allow customers requirement to be discussed in details for easy understandable of what kind of problem customers are trying to solve and then a solution mock-up is agreed on, hence the whole system will not be built, if the customer sees no value.

> "*Early and open customer engagement to understand what problem we should solve , mock-ups and prototyping before starting to code the full solution.*" (Technical Coordinator, CQ-2)

Our respondent elaborated that they are producing very detailed, easy to understand and standard documents describing how features implementation should be, iteration to make, and how the phase has been executed and done. After that, the documents are shared to the development teams and the whole process into the moment a feature or requirement has been developed in the code. This document should be reusable. Hence Hierarchy-based strategy could be used to addresses effects of architectural design and increased waste.

> "*Try to create and make standard documents or templates which can be reused for every kind of project.*" (CI/CD Consultant, CY-1)

While this is true that DevOps practitioners use different strategies, these strategies have their pros and cons. To remedy these effects, it's imperative that alternative strategies, in terms of the practices and roles involved, can be outlined as general solutions rather than only the discussed one.

VI. Discussion

The findings of the study reveal that the identified strategies are characterized by a top-down approach. This means a high-level requirement is first provided and then decomposed into smaller stories, cases or epics. This Figure 3, depicts those strategies. This section will further discuss the highlighted effects and suggest alternatives to mitigate them. Our suggestions would not disturb the overall strategy rather than mitigate potential risks.

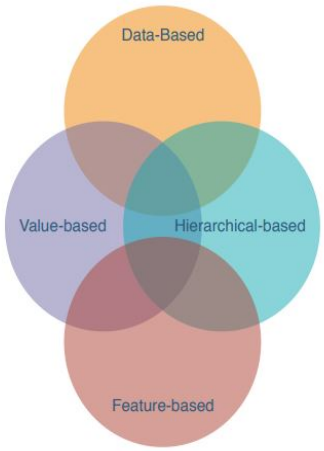

Fig. 3. Strategy Overview

The implementation of DevOps did not provide the required agility in analyzing and breakdown requirement in the case of company Q. In that context, The process is still driven by the complexity of the software, thus, it limits the DevOps capabilities to achieve an optimized level of DevOps maturity [30]. The strategy identified in case Q poses several challenges persistent form agile requirements solutions which will be listed in a latter part of this study. In a relatively less critical system, the contrast was not visible between feature-based, data-based, value-based strategies and literature about agile requirement engineering [14]. Agile overcame traditional software development issues and provides agility in developing software. One of the agility is primarily achieved by the ability to handle requirements change. The requirements analysis and breakdown is tailored to favour incremental releases to customer. This is done by breaking down software requirements into user stories [14]. DevOps and agile are synchronized to release software faster to the customer. The value behind DevOps lays in bridging between departmental functions.

Pre-existing agile strategies can still be used in agile-DevOps settings. However, the effects of those strategies are still persistent form agile way of analyzing and breaking down requirement, and has not been addressed in the industry.

A. Delivery Speed

In a hierarchical, the time required for requirement preparation is regarded high by our interviewee and a new way of working with requirement analysis is required to optimize software delivery, thus, get faster feedback from the customer.

> "*I would say maybe two years from the moment that trigger the feasibility study until we have it done in production. The product got, of course, I tried to make it faster, but this is a pure waterfall process that we haven't changed so far after we have gone agile and we have introduced DevOps, we need to work like this.*" (Programme Coordinator, CQ-1)

Unlike the other strategies, in the hierarchical-based, this is aggravated by a high number of hand-offs in order for a requirement to be deployed. The concept of autonomous teams and required skills in a DevOps environment is limited as no indication about a capacity to cooperate, between requirements teams, development, and operations [16]. This affects one of the key attributes of DevOps attributes "OpsinDev" [13]. The role of the operational teams is limited in the analysis and breakdown process. The strategies disentangle the process from multiple expertise. However, It is essential for Devs to understand the real-world production environment where Operational teams release the code. In a similar way, operational teams need to collaborate with the developers and configure the system according to how the code is being produced. Furthermore, in a DevOps environment, requirements are not only shaped by development feedback but also by operations monitoring. Hence, operational teams much know which requirements are being developed. The incorporation of operational teams in the process will add agility to the overall process and help product owners address requirements issues faster int the production environment.

Moreover, there should be a balance between continuous engineering practices and change management across all cases based on Mohamed's work [30]. The change management in the case of our research, involves the requirements change. This applies to the hierarchical strategy. This strategy offers a better requirement specification and traceability. Contrarily, to the rest of the strategies, which usually causes rework on requirements, thus impede continuous integration by increasing the time from requirement analysis until the integration of a task. In feature and value-based strategies, cross requirements management has been reported to cause undesirable delay and lead to failed tests. In value-based strategy, this aspect is limited by the capabilities of the tools used for task management to handle the requirements inter-dependencies. A lack of tools to handle requirements conflicts.

Suggestion: Cross-requirements create dependencies on deliveries thus it will be useful to discuss early requirements conflicts and align the tests with requirements. This can be supported in the development by tools for requirement management in large scale software development. Mismanagement in requirement can cause errors in the system integration tests thus delay software delivery. Feature-strategy gives better delivery speed but has much work to resolve conflict/dependencies, however, a more modular architecture such as microservice would address the latter. Microservice architecture [5] supports autonomous teams which is emphasied in the feature-strategy.

B. Architectural Design

Mapping requirement breakdown with software architectural is challenging task [7], this affects the design choices and the overarching architectural style. This is lined to Case X as they said having multi-layered decomposition cause misunderstanding from architectural perspective.

Suggestion: One potential suggestion is the requirement traceability and decomposition should work hand in hand. Also, early participation of the testers in the requirements work. This is observed in the feature-based and data-based strategies, which offer better support incremental improvement to architecture In feature-based and data-based strategies better support incremental improvement to architecture.

C. Requirements Process

This section reflects on the implication on the remaining requirements engineering activity as well as the requirements process output.

Feature and value based strategies did not have neither an established approach for traceability nor a standardized way of requirement specification which may impact the delivery process [4]. This also increase rework on requirement which is typically handled in meetings and forums in both strategies. This aspect is yet to be digitilized which can facilitates the participation of a larger number of teams .

In feature-based and value-based strategies, development teams often end up with missing information in the backlogs. This is caused by under-specified requirement or missing intrinsic dependencies links between requirements [45]. Similar issues get readdressed in frequent meetings and architectural forums. However, this rework impedes lead time and require more resources from the company which is typically overlooked. Thus, to deal with the and cross-system requirements, in the hierarchical strategy peoples use their personal network to learn about potential conflicts from other teams. This is oblivious as one of our interviewee mentioned "*I'm a bit against the actually putting too much time in breaking down because it takes a lot of time and it's the often, it's not until we start working on something that you really understand what do you need to do? its sort of hard to break something down too much from the beginning..*" (CI/CD DevOps Analysts, CZ-1).

Moreover, breaking down requirements of to small chunks of business value is not only reported in [20], but also has been found problematic as one of the interviewee replied

> "*Try to better keep out small issues that have a tendency to get into the backlog that is not directed at value creation.*" (Product owner, CP-1)

. Challenge regarding breaking down knowledge about customer value is reported by the researchers.

Suggestion: Validate proposed solutions for requirement analysis and cross-communicate across roles. The solution can be in the form of a set attributes such as testability, integrability specific to each level of abstraction that should be verified in the requirement hierarchy of the case. This is observed in the hierarchical based strategies, in which , more resources and concrete techniques are provided to the requirements analysis and breakdown process. However, driven by the fast delivery pace and high testing frequency, requirement analysis should not only identify the dependencies between requirements but also to communicate them. Hence, this can be facilitated by involving multiple expertise in this process.

Suggestion: One major impact is having multiple requirements authors, Reducing the requirement structure size and handoffs, and allow more developers to be involved in this process however concerns have been raised regarding the level of knowledge of the system as this seems to be challenging for large scale systems.

Finding from this study shows that case Z relied more on their experience and their intuition on breaking down requirement. Implicitly, they favour people and interaction over a defined process. It is not easy to understand whether small changes that do not directly add value to a feature are worth integrating or not.

D. Testing

No strategies indicated a positive change in the requirement breakdown to supported a high number of test execution. It is imperative for the case company to achieve faster delivery of software daily to the main track. However, continuous delivery requires various types of tests to be automated. Initially, CI must be backed up by a concise and clear requirement breakdown [27]. Respondents from feature-based strategy reported that determining how much a single requirement must be broken down is a challenging task and should align with the system architecture.

Suggestion: Testers work should be familiar with the analysis and breakdown phases in order to which tests cases should be generated at which level of abstraction. Linking testing to requirements can be thought of as a practice. Continuous integration, delivery, deployment requires high number of different types of tests to be executed daily.

E. Autonomous teams

Across all cases apart from case Q has dedicate operational teams. The operational teams are discarded from the requirements process thus this limits their knowledge about some aspects of the development lifecycle such as release planning.

We observed that the aspect of the autonomous teams is limited. Top hierarchies in the process are engaged in the requirements analysis and breakdown process. They reflect and posits that this area particularly requires and extensive knowledge of systems and subsystems. Therefore, we observed that agility in the requirements is still hindered by the level of knowledge that DevOps practitioners need to possess. Specifically for (Case Q), driven by system complexity and a large number of components and subsystems, little is left to the development teams. This to some extent limit the capabilities to leverage the effort and skills of the cross-functional teams. Also multiple requirements author make analysis risky increase information and adversely affect the accuracy of the analysis [44]. Building on these needs, companies need to develop concrete planes to address their DevOps capabilities and invest on their resources skill set to cope with this new style of IT. Those improvement needs also to be on the requirements analysis level.

Suggestion: For the purpose to increase the capabilities of the cross functional in the hierarchical strategy, it is required to merge the requirement systematization department with developers in order to ensure reliable software releases within shorter time.

IN both feature-and value based strategies, Devs interfere in the requirements analysis at a lower level. However, we report on a gap in the knowledge required to decompose requirement. In Feature-based and Value based strategies requirements is decomposed based on speculation and assumptions rather then predefined . **Suggestion**: We suggest that on a project level the teams should include training session and workshops and agree on one common technique for analyzing and specifying requirements level on the team's or department level, give further support to the concept of autonomous teams in DevOps environment favour the autonomous teams.

The findings of this study did not showcase how DevOps is different in agile-DevOps environment. From the perspective of requirement analysis and breakdown challenges seems to

persistent form the agile way of breaking down software requirements. Strategies do not follow standardized techniques in this process to predict potential risks. Instead, they wait for the issues to arise which in most cases addressed through meetings rather than formal documentation traceability and updates.

VII. Conclusion & Future Work

In this study, we explored in a multiple case study of how requirements analysis and breakdown is conducted within five distributed cases of agile-DevOps environment. Further this study reports the effects and on DevOps implementation practitioners. The incorporation of DevOps in an agile environment aims to bring software faster to the customer. However, this fusion did not produce any changes on requirements process level most notably on analysis and breakdown. This still presented challenges to DevOps practitioners which part of it is inherited form agile. The effects found hindering a faster software delivery to the market were requirement process related, especially, assumption and speculation over requirement. Also on architecture design, lack of shared vision within the team, release and deployment. In conclusion, it seems that in the industry, especially with regards to the adoption of DevOps, DevOps practitioners did not accommodate refinements to the requirement analysis and breakdown, which some effects highlighted in this study proves the opposite. This study further suggest improvements to the strategy identified for the aim to enhance software delivery speed. these improvements targets the practices used within the strategies and how can be tailored to support DevOps attributes. Based on our results collected, we highlight one area that we believe that DevOps researchers can contribute to:

- A cross-analysis of aspects of development and process elements when developing in house and outsourcing DevOps tools in organizations that have adopted DevOps.

Such contribution will foster perpetual automation and faster delivery for organizations while providing support to their business development teams by developing in house tools and infrastructure pipelines in order to adapt to the target development pace. Therefore future work would benefit in evaluating and comparing to what extent requirements engineering practices and DevOps differ in an agile environment using similar domains used in our research. Further, we hope our results alleviated vagueness regarding how operational requirements are analyzed in Agile-DevOps setup, thus trigger further researcher in how functional and operational requirements management and specification is conducted.

Acknowledgements

Foremost we want to thank our families for their unconditional support. We also want to thank all the interviewees in the study for their valuable input during the interviews. We also thank CI/CD consultant & developer from Business IT Nerd for his help in providing thesis feedback, relevant company related information to connect both the theory and practical aspects. Lastly our supervisor Lucy Lwakatare for her invaluable feedback and guidance throughout this thesis.

References

[1] Julian M Bass. How product owner teams scale agile methods to large distributed enterprises. *Empirical Software Engineering*, 20(6):1525–1557, 2015.

[2] Len Bass, Ross Jeffery, Hiroshi Wada, Ingo Weber, and Liming Zhu. Eliciting operations requirements for applications. In *Proceedings of the 1st International Workshop on Release Engineering*, pages 5–8. IEEE Press, 2013.

[3] Elizabeth Bjarnason, Michael Unterkalmsteiner, Markus Borg, and Emelie Engström. A multi-case study of agile requirements engineering and the use of test cases as requirements. *Information and Software Technology*, 77:61–79, 2016.

[4] Elizabeth Bjarnason, Krzysztof Wnuk, and Björn Regnell. Requirements are slipping through the gapsa case study on causes & effects of communication gaps in large-scale software development. In *2011 IEEE 19th international requirements engineering conference*, pages 37–46. IEEE, 2011.

[5] Lianping Chen. Microservices: architecting for continuous delivery and devops. In *2018 IEEE International Conference on Software Architecture (ICSA)*, pages 39–397. IEEE, 2018.

[6] Jürgen Cito, Philipp Leitner, Thomas Fritz, and Harald C. Gall. The making of cloud applications: An empirical study on software development for the cloud. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2015, pages 393–403, New York, NY, USA, 2015. ACM.

[7] Adam Debbiche, Mikael Dienr, and Richard Berntsson Svensson. Challenges when adopting continuous integration:: A case study. volume 8892, pages 17–32, 2014.

[8] Ulf Eliasson, Rogardt Heldal, Eric Knauss, and Patrizio Pelliccione. The need of complementing plan-driven requirements engineering with emerging communication: experiences from volvo car group. In *2015 IEEE 23rd International Requirements Engineering Conference (RE)*, pages 372–381. IEEE, 2015.

[9] Floris Erich, Chintan Amrit, and Maya Daneva. Report: Devops literature review. *University of Twente, Tech. Rep*, 2014.

[10] Brian Fitzgerald and Klaas-Jan Stol. Continuous software engineering and beyond: trends and challenges. In *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering*, pages 1–9. ACM, 2014.

[11] Brian Fitzgerald and Klaas-Jan Stol. Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*, 123:176–189, 2017.

[12] Tony Gorschek and Claes Wohlin. Requirements abstraction model. *Requirements Engineering*, 11(1):79–101, 2006.

[13] Viral Gupta, PK Kapur, and Deepak Kumar. Modeling and measuring attributes influencing devops implementation in an enterprise using structural equation modeling. *Information and Software Technology*, 92:75–91, 2017.

[14] Ville T Heikkilä, Daniela Damian, Casper Lassenius, and Maria Paasivaara. A mapping study on requirements engineering in agile software development. In *2015 41st Euromicro conference on software engineering and advanced applications*, pages 199–207. IEEE, 2015.

[15] Ville T Heikkilä, Maria Paasivaara, Casper Lasssenius, Daniela Damian, and Christian Engblom. Managing the requirements flow from strategy to release in large-scale agile development: a case study at ericsson. *Empirical Software Engineering*, 22(6):2892–2936, 2017.

[16] Aymeric Hemon, Barbara Lyonnet, Frantz Rowe, and Brian Fitzgerald. From agile to devops: Smart skills and collaborations. *Information Systems Frontiers*, pages 1–19, 2019.

[17] Jez Humble and David Farley. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Pearson Education, 2010.

[18] Irum Inayat, Siti Salwah Salim, Sabrina Marczak, Maya Daneva, and Shahaboddin Shamshirband. A systematic literature review on agile requirements engineering practices and challenges. *Computers in human behavior*, 51:915–929, 2015.

[19] Teemu Karvonen, Woubshet Behutiye, Markku Oivo, and Pasi Kuvaja. Systematic literature review on the impacts of agile release engineering practices. *Information and Software Technology*, 86:87–100, 2017.

[20] Rashidah Kasauli, Grischa Liebel, Eric Knauss, Swathi Gopakumar, and Benjamin Kanagwa. Requirements engineering challenges in large-scale agile system development. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 352–361. IEEE, 2017.

[21] K. Louise Barriball and Alison While. Collecting data using a semi-structured interview: a discussion paper. *Journal of Advanced Nursing*, 19(2):328–335, 1994.

[22] R. R. Lutz. Analyzing software requirements errors in safety-critical, embedded systems. In *[1993] Proceedings of the IEEE International Symposium on Requirements Engineering*, pages 126–133, Jan 1993.

[23] Lucy Ellen Lwakatare, Teemu Karvonen, Tanja Sauvola, Pasi Kuvaja, Helena Holmström Olsson, Jan Bosch, and Markku Oivo. Towards devops in the embedded systems domain: Why is it so hard? In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 5437–5446. IEEE, 2016.

[24] Lucy Ellen Lwakatare, Pasi Kuvaja, and Markku Oivo. Dimensions of devops. In *International conference on agile software development*, pages 212–217. Springer, 2015.

[25] Lucy Ellen Lwakatare, Pasi Kuvaja, and Markku Oivo. An exploratory study of devops extending the dimensions of devops with practices. *ICSEA 2016*, 104, 2016.

[26] Lucy Ellen Lwakatare, Pasi Kuvaja, and Markku Oivo. Relationship of devops to agile, lean and continuous deployment. In *International Conference on Product-Focused Software Process Improvement*, pages 399–415. Springer, 2016.

[27] Torvald Mårtensson, Daniel Ståhl, and Jan Bosch. Continuous integration impediments in large-scale industry projects. In *2017 IEEE International Conference on Software Architecture (ICSA)*, pages 169–178. IEEE, 2017.

[28] Nigel Mathers, Nick Fox, and Amanda Hunn. *Using Interviews in a Research Project*, pages 113–134. 01 2000.

[29] Michael MLewis-Beck, Alan Bryman, and Tim Liao. *Multiple Case Study*. 2004.

[30] S Mohamed. Devops maturity calculator domc-value oriented approach. *International Journal of Engineering Science and Research*, 2(2):25–35, 2016.

[31] Helena Holmström Olsson, Hiva Alahyari, and Jan Bosch. Climbing the" stairway to heaven"–a mulitiple-case study exploring barriers in the transition from agile development towards continuous deployment of software. In *2012 38th euromicro conference on software engineering and advanced applications*, pages 392–399. IEEE, 2012.

[32] Nicolás Paez. Versioning strategy for devops implementations. In *2018 Congreso Argentino de Ciencias de la Informática y Desarrollos de Investigación (CACIDI)*, pages 1–6. IEEE, 2018.

[33] Tuomas Paulin. Devops in finland-study of practitioners' perception. 2018.

[34] Richard A Powell and Helen M Single. Focus groups. *International journal for quality in health care*, 8(5):499–504, 1996.

[35] Z. Racheva, M. Daneva, and L. Buglione. Supporting the dynamic reprioritization of requirements in agile development of software products. In *2008 Second International Workshop on Software Product Management*, pages 49–58, Sep. 2008.

[36] Björn Regnell, Richard Berntsson Svensson, and Krzysztof Wnuk. Can we beat the complexity of very large-scale requirements engineering? In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 123–128. Springer, 2008.

[37] Leah Riungu-Kalliosaari, Simo Mäkinen, Lucy Ellen Lwakatare, Juha Tiihonen, and Tomi Männistö. Devops adoption benefits and challenges in practice: A case study. In Pekka Abrahamsson, Andreas Jedlitschka, Anh Nguyen Duc, Michael Felderer, Sousuke Amasaki, and Tommi Mikkonen, editors, *Product-Focused Software Process Improvement*, pages 590–597, Cham, 2016. Springer International Publishing.

[38] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14(2):131, 2009.

[39] Per Runeson, Martin Host, Austen Rainer, and Bjorn Regnell. *Case Study Research in Software Engineering: Guidelines and Examples*. Wiley Publishing, 1st edition, 2010.

[40] Mojtaba Shahin, Muhammad Ali Babar, and Liming Zhu. Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. *IEEE Access*, 5:3909–3943, 2017.

[41] Hossein Sharifi and Z Zhang. Agile manufacturing in practice-application of a methodology. *International Journal of Operations & Production Management*, 21(5/6):772–794, 2001.

[42] E. J. Uusitalo, M. Komssi, M. Kauppinen, and A. M. Davis. Linking requirements and testing in practice. In *2008 16th IEEE International Requirements Engineering Conference*, pages 265–270, Sep. 2008.

[43] Stefan Wagner, Daniel Méndez-Fernández, Marcos Kalinowski, and Michael Felderer. Agile requirements engineering in practice: Status quo and critical problems. *CLEI Electronic Journal*, 21(1), 2018.

[44] Jonas Widerberg. Bam-backlog assessment method. *Agile Processes in Software Engineering and Extreme Programming*, page 53.

[45] Hasan Yasar. Implementing secure devops assessment for highly regulated environments. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, page 70. ACM, 2017.

[46] S Magnus gren, Eric Knauss, Rogardt Heldal, Patrizio Pelliccione, Gösta Malmqvist, and Jonas Bodén. The manager perspective on requirements impact on automotive systems development speed. In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pages 17–28. IEEE, 2018.

VIII. Appendix

APPENDIX A
INTERVIEW QUESTIONS

Interview questions	Category
1) What is your current role and how many years of experience do you have in this role and in DevOps in general? 2) What is the context and scale of the project that you are working on?What is your team responsible for? What is the current hierarchy of your team? Which stakeholders you can constantly working with? 3) Can you give a high level description of your individual work flow? 4) Can you briefly describe the overall work flow in your work environment? • How does a typical requirement analysis process look like? • How does a typical delivery/release cycle look like? • Testing environment for functional and nonfunctional requirement	**Background**
RQ1:*What strategies adopted by DevOps practitioners for requirements analysis and breakdown in practice?*	
1) Who is the source of your requirement? And in what form are they given? 2) Which team (unit/person/roles) is responsible for analysing and breaking down requirements? which roles are involved? 3) What tools are used to decompose requirements? Do you adjust the requirements according to (external) changes after when the requirements are set-up? If yes, how do you deal with it? 4) which artifacts and on which level do you produce from the requirement analysis and breakdown? Do you provide those artifacts to anyone? 5) Are requirements provided in different abstraction levels to different teams? How so? How abstract the requirements you receive are? 6) How many iteration do you it is necessary to fully integrate a requirement? 7) How a task/story is fit for development? Are there criteria that a single requirement must verify 8) When and how are the system non-functional requirements determined? How are they monitored? 9) Are operational teams involved or have an affect or in the requirements process operational teams? • Do you think there are gaps in the information provided? • Do you think receiving requirements from various sources or having multiple iteration in the RE process effect requirements decision making? • The communication of requirement is it easy to trace the different levels of a single requirement?	**Strategies**

RQ2:*What effects do the current practice strategies present for DevOps practitioners?*	
1) What challenges do you face when analysing and implementing the provided requirements? and how do you solve those? 2) Given your current project scale, what is the estimated time for the effort put into materializing tasks? and what is the average time elapsed from integration until production 3) Based on your experience, what effect does the requirement analysis and breakdown process have on: • Early-design decisions • Continuous integration and delivery • Automated Testing • Integration Testing • Continuous delivery • Frequent release • Operational requirements? 4) How your requirement analysis and breakdown process work with continuous integration? How often have your team faced a problem such as having higher active branching tasks as compared to the number of active development.	**Challenges**
RQ3:*What alternative strategy could be used to overcome the challenges and accelerate software delivery?*	
1) In requirement analysis context, what would you do differently to reduce customer time-to-value? 2) In your opinion, does it exist any bottlenecks in the requirement breakdown process?. 3) Which roles do you think also should be involved in the requirement and analysis process?. 4) Though all your experience working in DevOps environment, How your requirements analysis and breakdown process evolved ? are you of any previous bottlenecks that you solved in other projects? • Do you think of any suggestions that could potentially improve the current ways of requirement analysis. • What Lessons have you learned from your requirements analysis process improvement?	**Alternative Strategies**

APPENDIX B
NVIVO NODES

Alternatives	0	0
Analysis approach	2	2
Autonomous Teams	2	3
Increase Domain Knoweldge	2	3
Prototyping	1	1
Reusability	1	1
Roles Involvement	2	3
Secure Customer Value	4	6
Shared View	1	2
User Experience	1	1
Challenges&Effects	0	0
Acceleraton of Software Delivery	4	11
Architectural Design	4	5
Mindset	2	2
Requirement Process	4	14
RequirementsToIntegration	2	2
Shared Vision	3	14
Feature Developement Conflict	1	1
Increased Waste	3	8
Team Coordination	3	5
Testing	3	9
encourage	1	1
Strategy	1	3
CI&Testing	4	20
Goals	2	13
Practice	5	44
Req Abstraction	5	11
Req Breakdown	4	19
Roles&Responsibilities	4	23
Tools	5	11

Reference 1 - 0,73% Coverage

And you know, you need to have a fairly good number of coordination between people or in such large context and who are sometimes working against each other. And they need to know about that. They need, I mean some, I mean that's why for example, one way to address that, let's say the system can be programmed that only that it's a feature x is enabled feature . y can not be enabled. You get an error message saying that you have a feature. X so feature y would not be enable to refer to that document for information, but you need to have high level of synchronization between teams and component people working in the development flow in order to realize that

- § 2 references coded [0,80% Coverage]

Reference 1 - 0,53% Coverage

to be efficient there because, there are a lot of the teams here and in different kinds of companies or different, departments.. but I think , it's mostly up to the product owners. They had some, they had meetings or a structure for discussing what they are doing now and what kind of, what they were focusing on, but that, but happens, that things get built twice or three times.

Reference 2 - 0,27% Coverage

because it's also requires some, a lot of the communications between teams, and on this high level for that, , high level, requirements. I guess you also had to have some kind of clear vision

- § 2 references coded [0,86% Coverage]

Reference 1 - 0,51% Coverage

But one of the biggest challenges is that on large teams from the connected services where i come from is we have high focus on creating autonomous teams that should work autonomously and not to speak so much with other teams for example, except for the forums that I mentioned so we have seen an effect of that which some team may be reinventing the same wheel multiple times

Reference 2 - 0,36% Coverage

when the teams are encouraged to work very autonomous could also be harsh miscommunication between the teams when they are parts that are interconnected in terms of a system that the end user faces so the miscommunication is also one thing that is challenge for us

APPENDIX C
HIGH LEVEL THEMES