Universiteit Leiden

ICT in Business and the Public Sector

Introducing Capabilities to a DevOps Maturity Model

Name: Tarık Emin Kaplan
Student-no: 3287254

Date: 29/08/2023

1st supervisor: Dr. C.J. Stettina MSc
2nd supervisor: T.D. Offerman MSc

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

DevOps is a popular paradigm in organizations delivering software, calling for organizations to unite their development and operational work and use automation substantially, in order to increase their delivery performances [1], [2], [3].

Decades of research has highlighted the significance of organizational capabilities and routines [4], [5], [6], [7]. Despite that fact, the DevOps maturity models created in order to help organizations assess their DevOps maturity have not been developed with a focus on this topic [8], [9].

This research studies the problems associated with DevOps maturity models, how the perspective of organizational capabilities and routines relate to DevOps, and how this perspective could be utilized in order to address the problems relating to DevOps maturity models. In order to do this, a first round of interviews was conducted with 10 DevOps practitioners. Following these interviews, one of the DevOps maturity models used in practice was improved and expanded with the lens of organizational capabilities and routines. A second round of interviews with 11 DevOps practitioners, 8 of whom participated in the first round, was conducted to validate this improved model.

The findings indicate that the model is moderately successful in helping practitioners understand which capabilities and routines their organizations need to adopt or improve in order to increase their maturity; and is easy to use (which is shown to be a very important factor).

However it was also shown that the link between the original model and the capabilities featured in it is not well established; the model may not be covering a complete view of every capability; and it may not be suited the best for DevOps practitioners who are doing DevOps using "alternative" team structures.

There are also other findings that could be interesting to both the practitioners and the researchers. For the latter, it is found that the capabilities and practices related to the "Cultural" dimension of DevOps do not necessarily act as a foundation for other dimensions; instead, all the dimensions are related to one another and the practices related to them are meant to be adopted together. For practitioners, it is found that the maturity assessments lead to practical improvements only when the assessment is done in regular periods, and when a strong focus on improvements from the last time the assessment was performed is present.

Acknowledgement

I would like to thank my supervisors Tyron Offerman and Christoph Stettina for guiding me through this journey, your help and flexibility in order to make this research happen was greatly appreciated. I also would like to thank all the people who have participated in the interviews as part of the research, though I cannot list your names to keep your anonymity, you all know who you are, and I appreciate your effort to contribute to this project. In particular, a special thank you to R.B.; from contributing to the interviews, to connecting with me with other people for more interviews, and for your advice when I reached out; your help for this research has been invaluable. Most importantly, I would like to thank my family as well, if I were to write down the reasons on how grateful I am for having you all, it would be longer than this paper.

Contents

Chapter 1

Introduction

In the current landscapes of businesses in nearly all industries, adopting information technologies (IT) and software has been, and is continuing to be an integral part of their value chain; as it brings significant economic benefits to organizations, governments, and consumers alike [10]. This mass adoption of IT and software is commonly referred as "digitization". Digitization requires organizations to invest in new technologies, and one of the most popular method regarding such investments is to create these new technologies by orchestrating software development within organizations [11].

Maturity models are popular with regards to many aspects of software development (e.g. project management, business processes), both in terms of use and in terms of creation by individuals and organizations [12]. The intent behind the creation of maturity models is to define an "anticipated, desired, or logical path" from an initial state to a mature state, where there are typically a "sequence of levels (or stages)" included to define such a path [13]. Maturity models can help organizations "identify organizational strengths and weaknesses while also providing benchmarking information at the same time" [12].

Traditionally, software development has been done across silos, two of these silos being development and operations. Years of software development done in this "traditional" way has resulted in development and operations teams within the same organizations having different understanding of problems, and even culture [1]. To bridge this internal separation and to deliver high quality software continuously, DevOps was introduced in the last decade [14].

DevOps finds its roots within Agile and Lean, both of which define certain methodologies, ways of working and a mindset regarding software development, and both are remaining to be used widely by practitioners today [14], [3]. Agile primarily focuses on incremental delivery, team collaboration, continual planning, and continual learning; and as a result aims to achieve high delivery performance and flexibility in handling customer inputs [15], [16]. Lean also has a similar focus, but it puts a large emphasis on the concept of "eliminating waste"; which explains to remove activities (expressed in monetary terms or time) that don't add value to the customer [17], [18].

DevOps is a paradigm calling for the "establishment of cross-functional, agile teams that are responsible for development and operations of their systems", and automation is a key part of DevOps [1], [2]. DevOps has been gaining popularity, both in research and in practice; but for the latter, it is especially popular within organizations where digital transformation is taking place; to change the ways of working associated with the aforementioned "traditional" software development, which is now outdated [2], [19], [20].

The "Dev" in DevOps refers to the development team, which is responsible for creating

and modifying software products; while the "Ops" refers to the operations team responsible for the deployment and maintenance of such products. The routines of each team entail duties in line with this description but it is absolutely necessary for these two teams to collaborate and communicate for optimal execution of DevOps as a strategy [21]. Typically, this is achieved by organizing team(s) such that practitioners belonging to both the Dev and the Ops disciplines are on the same team(s), but there are also alternative team structures when implementing DevOps [22].

The increasing use of DevOps has resulted in the birth of DevOps maturity models, which are created by organizations and by researchers alike. However, the DevOps itself is a relatively "young" paradigm within software development, as a result of this, there are different interpretations associated with the organizational capabilities and routines related to DevOps, which is also reflected within different DevOps maturity models [23], [9]. In other words, DevOps maturity models themselves are not at a "mature state". Furthermore, there aren't too many of them compared to other aspects of software development [9].

Organizational capabilities (also referred simply as "capabilities" throughout the paper) and routines - and their popular sub-topic, dynamic capabilities - all have been a popular subject of research within the academics for decades, and the topic is considered to be at a fairly mature state, as the researchers and practitioners have reached a relatively common understanding [4], [5], [6], [7]. On the other hand, the capabilities and routines (including dynamic capabilities) in the face of DevOps have only recently been beginning to be explored in literature [8], [2]. This indicates that the DevOps maturity models already developed have been likely done so without much consideration to the perspective of organizational capabilities and routines.

Therefore, it is the objective of this research to understand more about the capabilities and routines associated with DevOps, and to reflect this understanding within a DevOps maturity model.

1.1 Document Structure

The remainder of this paper is as follows: In chapter 2, the key pieces of literature reviewed for this research is explained in detail under three main categories: "DevOps", "DevOps maturity", and "Organizational capabilities and routines". In chapter 3, the problems identified within the literature is expressed, and the research question is introduced to further define the research objective. In chapter 4, the approach used within this research to answer the research question and the guiding questions is explained. In chapter 5, the insights gathered from the interviews, the maturity model created following these interviews, and the insights gathered validating the model are given. In chapter 6, the implication of the insights gathered are discussed in detail. In chapter 7, the paper is concluded with a reflection on the research question and potential future work that could be done to expand on this research.

Chapter 2

Literature

2.1 DevOps

One popular framework present within the DevOps literature used to define DevOps from an academic view is the CALMS (Culture, Automation, Lean, Measurement, and Sharing) framework; coined by Humble et al. [23], [21], [24]. CALMS is a framework that "assesses a company's ability to adopt DevOps processes, as well as a way of measuring success during a DevOps transformation", and it is a framework commonly used within the literature to describe the key components of DevOps [25]. Culture component emphasizes on the previously mentioned ideology that collaboration between "Dev" and "Ops" is necessary, and lays the foundation within an organization for DevOps to exist in the first place [26]. Moving away from project-oriented teams to product-oriented teams; establishing shared responsibility across teams and making them autonomous can foster such a culture [21], [27]. Automation component describes the need to replace the "manual error-prone repetitive tasks" with right tools [21], [27]. Doing so leads to faster execution of business processes, and enables establishing of many important practices associated with DevOps, such as continuous deployment, infrastructure as code, configuration as code [26]. DevOps practices find their origin in Lean and Agile, so the Lean component describes some key aspects of DevOps which are shared with Lean [3]. Namely, reducing work in progress (WIP) by reducing batch sizes is a principle of Lean, and it is also done so within DevOps by the means of breaking down software features and functionalities and shipping them individually [26]. Measurement component is about the "monitoring high-level business metrics [...] per unit time" [21]. Monitoring of such metrics are important to not only give the team key information about their processes, but to other stakeholders that may be involved with the software product(s) as well, which can help organizations make better and more informed decisions [25], [26]. Sharing component is tightly connected to the Culture component and it describes the importance of sharing across teams to break down business silos and promote collaboration [26]. Sharing knowledge across teams via tools encourage collaboration directly; while sharing of responsibilities brings teams closer with successful releases [21], [3].

There are two notable books related to DevOps: "The DevOps Handbook" [28] and "Accelerate" [3], both of which are co-authored by Jez Humble, the primary creator of the CALMS framework. "The DevOps Handbook", as it's name implies, describes and defines all that is DevOps. More specifically, it talks about "the theory, principles, and practices [practitioners] need to successfully start [their] DevOps initiative and achieve [their] desired outcomes" [28]. This is based on a wide collection of literature regarding technology and management, studies on organizations both done by authors and from literature, practical work done by authors within

organizations undergoing DevOps transformations, interviews done with relevant practitioners, and analyses of case studies [28].

In the "Accelerate" book, the authors analyse a significant amount of data collected over "many years" and using this data, they "measure and quantify software delivery performance, its impact on organizational performance, and the various capabilities that contribute to these outcomes" [3]. The capabilities are listed under 5 main categories: Cultural, Continuous Delivery, Architecture, Product and Process, and Lean Management and Monitoring. These two books turned out to be very valuable sources for this research, as it can be seen in later chapters.

Offerman et al. identified 14 of the most common DevOps practices [26], which are represented in table 2.1. These practices are in line with the aforementioned CALMS framework, so they provide a good foundation of knowledge to improve upon a DevOps maturity model.

#	**Practice**
1	Automated and continuous deployment throughout entire pipeline
2	Make small and continuous releases
3	Developers get feedback based on releases
4	Create development sandboxes for minimum code deployment
5	Everything is stored as code and under version control
6	Integrated configuration management
7	Automated and continuous testing in development and staging environments
8	Reduce the time it takes to test, validate and QA code
9	Code reviews are change based
10	Automated and continuous monitoring of applications and resources
11	Automated dashboards that include health checks and performance
12	Support configurable logging that can optionally be turned on/off as needed
13	Use trunk-based development over long-lived feature branches
14	Use Test driven Development where all code has unit tests

Table 2.1: Most common DevOps practices according to Offerman et al. [26]

2.2 DevOps maturity

According to Mettler, in the space of information systems, maturity is about measuring of the ability of an organization's progress in their selected discipline, towards "accomplishing of a target from an initial to a desired or normally occuring end stage"; and maturity assessment models (or just "maturity models") are tools to measure the stages such organizations are in [29]. According to our research of literature, this definition of maturity and maturity models can also apply to the DevOps maturity and DevOps maturity models. Following are the literature related to DevOps maturity and DevOps maturity models reviewed.

An article on DevOps practices, which serves as an "ancestor" for this study is the one by Offerman, Blinde and colleagues [26]. The article's main focus in on DevOps practices, but several key findings related to maturity are there in the form of practices correlations to maturity. The practices were already listed in 2.1. Out of the 14 practices defined by the authors, while all the

practices are positively correlated with DevOps maturity, practices such as “automated testing in environments”, “automated continuous deployments”, “automated and continuous monitoring”, and “configuration management” have the strongest correlations, whereas “sandboxes for minimum deployment”, “trunk-based development”, “trying to reduce time to test/QA” have the weakest. In this study, the perceived DevOps maturity of organizations were measured based on Eficode’s DevOps maturity model [30]. The authors justify this choice of maturity model by stating that it is in line with the CALMS framework. The thesis paper by Blinde preceding this article features 47 practices, among which the 14 practices featured in the article are also present [31].

The research done by Zarour and colleagues compares 7 DevOps maturity models based on their application, levels and dimensions; and the maturity model developed by Eficode is one of these models [9], [30]. The authors do recommend this model, stating it as a “comprehensive and promising” model, alongside the model developed by Bucena [32]. According to the authors, both models share important similarities in their dimensions and levels.

Eficode’s model (as shown in Figure 2.1), is a DevOps maturity model that features 7 dimensions (“Organization and Culture”, “Environments and Release”, “Builds and Continuous Integration”, “Quality Assurance”, “Visibility and Reporting”, “Technologies and Architecture”, and one unlabelled dimension on starting new projects), and 4 levels (numbered from 1 to 4, 1 being the lowest and 4 the highest) [30]. For each dimension/level pair, a basic description of how related activities are supposed to look like is given, and organizations are meant to pick the description that best fits them for each dimension, which results in their assessed level of maturity across 7 different dimensions. This means that it is possible to be at different levels in each dimension.

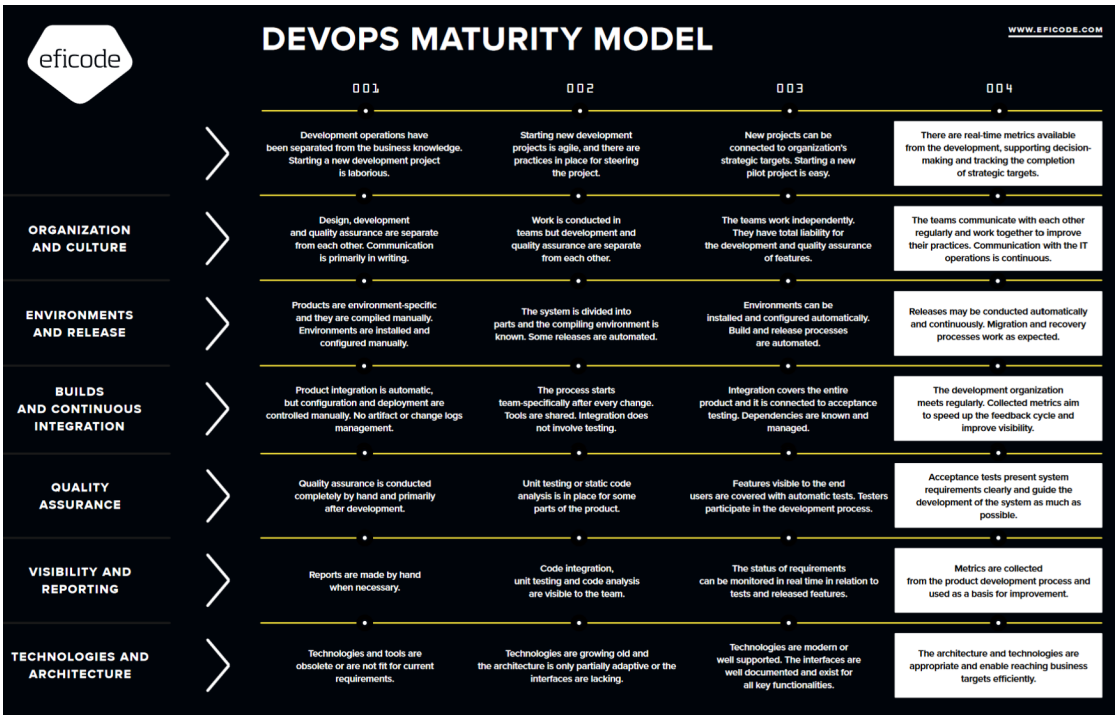

Figure 2.1: The Eficode’s maturity model [30]

One notable maturity model which was also featured in the paper by Zarour et al. was the one presented in the paper by Feijter et al. [33] [9]. It was the only DevOps maturity model we could identify which was explicitly based on capabilities. You can find the maturity model in Figure 2.2. This maturity model has 3 main categories (which are referred as "perspectives" in the paper), namely these are: "Culture and Collaboration"; "Product, Process and Quality"; and "Foundation". These categories can be thought as the equivalent of "dimensions" featured in Eficode's model. The authors came up with this based on a "DevOps competence model" they have developed, which in turn was created based on numerous literature and was further validated by expert interviews. Under each of these 3 categories there are some "focus areas", which can be thought as the DevOps practices organizations should possess. For each focus area there are a number of capabilities (represented by capital letters A,B,C and so on). The development of a capability within a single focus area depends on the existence of previous capability (i.e. there is no way to possess capability B without possessing capability A first), and the current state of a capability regarding each focus area determines the maturity "level" an organization is at for that particular focus area. It is possible to be at different levels across different focus areas.

The authors of this paper also highlight a key underlying problem with many DevOps maturity models which were based on Capability Maturity Model (CMM), such as the other maturity models featured in the paper by Zarour et al., and that is the fact that "no step by step guide is incorporated [within these models] that shows a growth path to become more mature" [9], [33]. While the maturity model proposed by the authors seems to be addressing this particular problem by the means of utilizing focus areas and capabilities; and is the only DevOps maturity model that actively feature capabilities according to our literature review, it has its flaws and doesn't address some other problems. We will be discussing about what these flaws and problems are in future chapters, and will also discuss about what could be utilized from this particular model and the paper associated with it, for the context of our research.

The article by Marnewick and Langerman gives valuable insights into the problems/benefits associated with DevOps maturity and current maturity models [34]. Through a case study, the authors reached the consensus that utilizing a DevOps maturity model does not increase organizational performance on its own, and it should be used in conjunction with other strategies. Furthermore they stated that "organizations are better off creating their own models based on the various models available". One important thing to note here is that the authors did indeed create their own DevOps maturity for their case study, but they also self-reported that their model did not really align with any of the 7 "mainstream" models referred by Zarour and colleagues, so their findings should be evaluated with caution [9]. Regardless, the authors do acknowledge that a maturity model was helpful (albeit, not sufficient) to drive organizational change; and that "various actions" did have a positive impact on the dimensions stated by their maturity model [34].

A literature review of DevOps and DevOps practices by Ali and colleagues gives a good review of some DevOps literature, and the authors also highlight some topics they identified based on the quantity of the literature they reviewed [35]. Firstly, they state the significance of the relation between DevOps and digital transformation paradigm; and state that the existing literature focuses on "tools, practices and the lack of standards", but there doesn't appear to be much focus on "adapting DevOps to an active IT ecosystem", and on adoption of some emerging technologies and what they could mean for DevOps. There is no mention of DevOps literature regarding DevOps maturity in this review, which may suggest that there is lack of such literature in comparison to other stated DevOps topics. The overall consensus of authors is that while DevOps is continuing to gain popularity both in practice and in research, at least for the latter, "its concepts and implementations are still cloudy, and doing its execution in routine is

Focus area \ Level	0	1	2	3	4	5	6	7	8	9	10
Culture and collaboration											
Communication		A				B	C			D	E
Knowledge sharing				A		B	C				D
Trust and respect							A	B	C		
Team organization		A	B						C	D	
Release alignment				A					B	C	
Product, Process and Quality											
Release heartbeat		A				B	C		D	E	F
Branch and merge			A	B		C		D			
Build automation			A	B		C					
Development quality improvement			A			B		C	D	E	
Test automation				A	B	C			D		E
Deployment automation					A	B		C			D
Release for production					A			B	C	D	
Incident handling			A					B	C	D	
Foundation											
Configuration management			A	B		C					
Architecture alignment			A					B			
Infrastructure				A			B	C	D		

Figure 2.2: The final maturity model developed by Feijter et al. [33]

not well-versed".

In summary, according to our review of the literature on DevOps maturity, it is safe to say that a gap in research exists regarding some topics. Namely, how the DevOps maturity assessment of an organization gathered by utilizing a DevOps maturity model should be translated into their practical work to improve their maturity, and exploring the potential usefulness of existing DevOps maturity models are such topics. Thus we will try to address these topics with this research.

2.3 Organizational capabilities and routines

Salvato's articles [36], [4] on organizational capabilities and routines provides valuable insights. One article is about the "microfoundation of routines and capabilities"; and the other is a research done under an organization to get a deep understanding of capabilities related to their new product development process, and the insights gained from it. According to Salvato's research, "microactivities" carried out by individuals (i.e. day-to-day routines) are central in shaping the core capabilities of organizations [4]. Furthermore, the term "capabilities" are sometimes used as a "black-box" phrase to describe organizations' abilities to enable firm-level outcomes; and routines are reduced to building blocks of capabilities [4], [36]. However Salvato argues that routines and capabilities have a more complex, fine-grained and multi-layered nature than that [36]. Salvato defines capabilities as "collections of routines characterized by evident firm-level purpose" and states that they enable organizations to "reliably perform and extend its characteristic output actions" [36].

Feldman and Pentland has made an important conceptualizing of routines by describing them from the perspective of two aspects, ostensive and performative [6]. The ostensive aspect denotes the descriptions of patterns (which may not always be written) that is used to carry out a task; while the performative aspect denotes the actual activities done by individuals at specific times and places in the name of carrying out said task. Authors argue that traditionally, the ostensive aspect has been widely assumed as the whole routine, which led to organizations putting too much emphasis on stabilizing their routines. Whereas the authors believe that understanding of this distinct aspects can lead to routines being leveraged as tools to drive change within organizations. This conceptualization of ostensive and performative aspects give more information to understanding of the aforementioned "multi-level" perspective of routines. This conceptualization is still used in current day literature.

Dynamic capabilities are no doubt an important consideration when talking about organizational capabilities. Teece defined dynamic capabilities as "the firm's ability to integrate, build, and reconfigure internal and external resources/competences[/capabilities] to address and shape rapidly changing business environments" [5]. Teece has long argued that dynamic capabilities are a source of competitive advantage [5]; and he shares the same belief with Salvato, in that he argues dynamic capabilities have a multi-layered nature [37]. In his recent article he specifically talks about how organizations with a strong dynamic capabilities reflect this strength in their business model; and conversely, how a good business model enables strong dynamic capabilities to be built [37].

A review of empirical studies on DevOps capabilities [8] and the "Design and validation of a capability measurement instrument for DevOps teams" paper [2] by Plant and colleagues gives great insights into many topics related to DevOps and capabilities related to DevOps (especially dynamic capabilities). There are many great findings, and here are some that can be considered important to this research:

- "Present DevOps maturity models are primarily focused on mapping capabilities to maturity levels, but did not investigate the potential ways in which a capability may be implemented" [2].
- The current DevOps maturity models does not appear to utilize organizational capabilities and routines to an in-depth level [2], [9].
- DevOps teams need to develop their dynamic capabilities on two levels: business-related and technology-related capabilities [2], [8].
- To achieve competitive advantage, DevOps teams need to develop some capabilities, given under 3 main categories: Organizational enabler capabilities, Dynamic DevOps team capabilities (capabilities belonging to this category are separated as business related or technology related), and DevOps outcome capabilities [8].

As a result of their work, Plant and colleagues have developed a "capability measurement instrument" [2]. In the process they have also defined a list of capabilities and practices associated with these capabilities, utilizing the CALMS framework while doing so. This list is the most comprehensive one (featured in a published article), we have identified regarding DevOps capabilities and practices. You can find this list in figure 2.3.

Another useful literature review is the one by Schilke and colleagues, which is on dynamic capabilities [7]. Authors believe that the research on the topic of "dynamic capabilities" has grown substantially and the topic has reached a certain level of maturity, since there is a substantial amount of empirical research and that the definition of dynamic capabilities remains

Capability	Description	Associated practices
Culture		
Intrapreneurship	The team has processes and structures in place to ensure it fulfills all necessary business activities in order to remain relevant to customers	Opportunity scouting Experimentation Problem recognition & solving
Continuous improvement	The team has processes and structures in place to ensure a team culture focused on communication and innovation	Accepting & providing feedback Continuous improvement Sharing goals & values
Self-empowerment	The team has processes and structures in place to ensure it can function independently without intervention from management	Change readiness Decision-making Self-organization
Automation		
Continuous software engineering	The team has processes and structures in place to ensure the continuous release of high quality software	Automated build Automated testing (Continuous) Integration (Continuous) Deployment
Infrastructure & configuration management	The team has processes and structures in place to ensure the necessary infrastructure is available and configured correctly	Infrastructure provisioning & containerization Managing configurations
Artifact management	The team has processes and structures in place to ensure artifacts are stored and versioned in a repository	Use of artifacts
Architecture management	The team has processes and structures in place to ensure the architecture is and remains flexible	Use of microservices or a modular architecture
Security & access management	The team has processes and structures in place to ensure their applications are secure, in line with compliance requirements and may only be accessed by authorized users	Performing risk analysis, risk evaluation, compliance requirements & security testing Using access policies
Lean		
Lean process management	The team has a process or framework in place to ensure optimum flow of work	Lean/Agile way of working Lean/Agile project management
Change & operations management	The team has processes and structures in place to manage change requests and systems operations	Resolving incidents Automated recovery Managing changes
Continuous planning	The team has processes and structures in place to ensure a flexible planning	Planning
Customer-centric design	The team has processes and structures in place to ensure their services are targeted at involving and meeting customer needs	Stakeholder management Product-oriented team setup Cross-functional team setup
Requirement management	The team has processes and structures in place to manage and prioritize system/service requirements	Requirement specification & prioritization Use of NFRs
Monitoring		
End-user monitoring & feedback	The team has processes and structures in place to ensure it is aware of how their system is used and improve it based on end-user behaviour and feedback	Monitor customer systems & receive feedback from end-users
System monitoring & documentation	The team has processes and structures in place to monitor the performance and behavior of their internal systems	Monitoring & logging of internal systems
Sharing		
Knowledge sharing	The team has processes and structures in place to ensure that information, knowledge and skills are equally distributed and disseminated throughout the team	Information sharing Continuous learning Sharing knowledge & skills
Team collaboration	The team has processes and structures in place to ensure regular alignment between team-members and with other teams in the organization	Intra-team alignment Inter-team alignment Sharing priorities

Figure 2.3: The list of capabilities and practices as featured in Plant et al. [2]

fairly consistent across literature (unlike DevOps literature as indicated by Ali and colleagues as mentioned earlier [35]).

Klievink and Janssen indicated that some stage models, which is a category of models most DevOps maturity models are also a part of, "does not include the dynamic capabilities needed by organizations to transform from one stage to the next stage" [38]. Though it's important to mention that this research in question was not related to DevOps, it did bring up some interesting questions related to our research. What if having a high DevOps maturity level (according to an established model) does not mean one ("one" as an organization) does not possess enough dynamic capabilities (and how can we define "enough" based on maturity?). Conversely, what if one does possess "enough" dynamic capabilities but does not have a high level of maturity? It is reasonable to assume that these could be possible, given the amount of research (or lack thereof) exploring DevOps maturity and dynamic capabilities, as pointed out within the research of Plant et al. earlier [8], [2].

Chapter 3

Problem Statement

The literature has shown us that, DevOps maturity models doesn't seem to be actively utilizing capabilities in order to lay down a path to achieve a certain level of maturity [2], [33], [38]. The only DevOps maturity model we have found that do utilize capabilities, focuses on mapping capabilities to certain maturity levels [33]. However, existing literature indicates the maturity models that do utilize capabilities doesn't even do so to an in-depth level; and that no maturity models investigate how capabilities could be implemented in practice [2], [8], [9]. Furthermore, the meaning behind the terms "capabilities" and "routines" are reduced to the ability of organizations to achieve a certain outcome, while some researchers argue that they have a much more complex, fine-grained and multi-layered nature [4], [36], [37]. There is also the consideration to dynamic capabilities. Even though the relationship between dynamic capabilities and DevOps have been explored in literature and is shown to be significant, according to our knowledge, there is no maturity model that features dynamic capabilities [2], [9].

Thus we recognize multiple problems. The first problem is that understanding what exact capabilities and routines organizations need and addressing improvement/development of such capabilities and routines may not always be so obvious following an assessment using an existing maturity model (this problem is shortly referred as the "actionability problem" throughout the paper). Analysing maturity models from the capabilities and routines perspective may help understanding this problem, and it may help organizations improve their capabilities and move from one stage to another in a given maturity model as a result; it would also make the link between the stages of maturity models and the capabilities and routines apparent. It may also help addressing the underlying "maturity models not being sufficient to drive organizational changes" problem stated by some researchers [34]. The second problem is regarding dynamic capabilities. As mentioned earlier, despite the importance of dynamic capabilities to DevOps, as highlighted in literature, there is a scarcity of implementations of dynamic capabilities within existing DevOps maturity models [2], [8]. This could result in potential disparities between the assessed DevOps maturity of organizations and their actual DevOps maturity, depending on the dynamic capabilities they possess. To resolve this, we need to explore how dynamic capabilities could be featured in maturity models; and our final maturity model needs to utilize dynamic capabilities.

3.1 Research question

As explained, within the realm of DevOps, maturity models have not been created with a thorough analysis of capabilities. As such, we believe it is important to see whether using a perspec-

tive that utilizes capabilities will yield different implications for DevOps, both in research and in practice. Therefore, the main research question (RQ for short) is expressed as follows:

"RQ: How can the lens of capabilities improve current DevOps maturity models?"

This main research question covers a large area of topics and is inherently difficult to answer on its own. Therefore, to help answering it, we will try to answer some guiding questions (GQ for short) which are less complex in nature, and will give us a narrowed down scope. This enables us to focus on certain topics, which when taken together helps us answer the main research question. The guiding questions are as follows:

"GQ1: What are the organizational capabilities and routines that are related to DevOps?"

Answering GQ1 will give us a thorough understanding of the organizational capabilities and routines associated with DevOps. This is to ensure a good foundational knowledge base is established before focusing on a maturity model to improve.

"GQ2: How do organizations move from one maturity level to another?"

Answering GQ2 will give us information on how we should design our model such that the information related to the capabilities and routines are in-parallel with the maturity progression of the model. Furthermore, such information should feature suggestions that would help organizations improve their capabilities and increase the maturity, and answering GQ2 can directly contribute to such suggestions.

"GQ3: How does a maturity model that utilizes organizational capabilities and routines fare in practice?"

After GQ3 is answered using the model created as a result of our research, an understanding of how practical the model is in solving the problems we have identified; and an understanding of how such a model can be improved will be established.

Chapter 4

Methodology

The research method we will employ to answer our research question is the design research. Design research is defined as "the study, research, and investigation of the artificial made by human beings, and the way these activities have been directed either in academic studies or manufacturing organizations" [39]. So, given the practical nature of DevOps and our research, it is appropriate to utilize this method, since we.want to not only design but also validate our proposed model.

As Salvato indicated in his research, to get a deep, multilevel understanding of routines and capabilities within an organization, it would be needed to observe data collected over a long period of time [36]. However, we don't intend to do that. The reasoning behind why we are not doing it is because we are not trying to unbox the concept of capabilities and routines, and we're only trying to understand how this concept would relate to DevOps maturity and maturity models. In other words, our research is in more line with focusing more on the aforementioned ostensive aspect of capabilities and routines rather than the performative aspect [6]. This is also the reason why we are using a design research methodology. We are seeking to get different perspectives of numerous experts from different organizations, rather than doing an extensive case study within a single organization.

As mentioned earlier, the maturity model developed by Feijter et al. was already made with capabilities in mind, and it was the only DevOps maturity model in the literature that we could find which was also proven by a case study [33]. We have found it appropriate to analyse this model in more detail and arrived at some findings associated with the model that could be problematic for our research. There are two main problems we have identified. The first problem is that the 3 categories (i.e. "perspectives"), the focus areas and capabilities may not be exactly in line with the aforementioned CALMS framework or the common practices identified by Offerman et al. [26]. Furthermore, instead of using an existing framework like CALMS as a foundation for their research, Feijter et al. compiled a list of "Drivers" of DevOps within organizations based on various literature and their interviews, and used that as their foundation [33]. You can see these drivers in figure 4.1. Some of these drivers appear to correspond to some of the components of CALMS, such as "Culture" and "Automation" but it can't be safely claimed that the model is in line with the CALMS framework or any other established framework used to describe DevOps. The second problem is that there is no consideration to dynamic capabilities at all. So, the maturity model created by Feijter et al. could be a good starting point, but it is incomplete for our purpose.

In addition to these two main problems, there also two minor problems. First is that, the authors have self-reported a large number of interviewees were employees of a single organization

Driver
A culture of collaboration
Agility and process alignment
Automation
Higher quality
Development and deployment of cloud based applications
Continuous improvement

Figure 4.1: The drivers used by Feijter et al.

(since this research was done under one organization); and since these interviews were used to validate and improve upon their maturity model, the final maturity model is tailored towards that particular organization and may not fit the need of other organizations [33]. The second problem is that the model is inherently a more complex and vastly different maturity model compared to the existing alternatives, which may cause the model to not be understood and used correctly by practitioners [9].

4.1 Interview Guide

We planned to conduct two cycles of semi-structured interviews with industry professionals experienced with DevOps. In the first cycle, we intended to focus on understanding what DevOps practices are executed, how DevOps maturity is currently viewed within organizations, and what the design requirements for a maturity model should be. You can see the set of questions we asked in the first cycle of interviews in Appendix A. After the first cycle of interviews ended, we have used literature and the insights we gathered from these interviews to make an informed choice with selecting a maturity model to improve on, and what those improvements could look like (these insights and improvements are talked extensively within future chapters).

In this first cycle of interviews we have primarily used the DevOps practices featured in the work of Plant et al. (figure 2.3) and secondarily the DevOps practices featured in the work of Offerman et al. (table 2.1), when asking questions to our interviewees about their practices.

After we had our improved version of the model, we planned to conduct a second round of interviews with a set of industry practitioners that included the same set of industry practitioners from the first interviews, plus some additional interviewees. In the second round of interviews we intended to assess the maturity of our interviewee's systems when we look at it with the lens of capabilities and routines to see how well our improved maturity model held up in practice, and whether it helped addressing the problems listed in our problem statement. To do this, we intended to evaluate our proposed model by the means of brainstorming with our interviewees around topics which are most relevant to the interviewee and the model (the exact set of questions are given in Appendix A). After the second cycle of interviews ended, we have analysed the results and presented our findings in the upcoming chapters.

We intended to collect both qualitative and quantitative data for this study, but we expected the former to outweigh the latter. The quantitative parts were anticipated to be related to get an assessment of DevOps maturity using the rating system used in our model. Qualitative parts were anticipated to be related to get more elaboration on the quantitative questions, and to get an idea of interviewees' routines and capabilities within their organizations.

4.1.1 Population

Given the intent behind the interviews is about learning about what the practitioners who do DevOps actually do within their respective organizations (i.e. their routines/practices), our main demographic for targeting potential interviewees are those that are actively engaged in building software and are following DevOps methodology to some extent, since such people can comment about what they actually do on a daily basis, and we will see how those comments relates to what we know about DevOps capabilities, routines and practices in theory that we have covered.

The people who fit that criteria typically identify as a "DevOps Engineer" (or a "Software engineer/developer" who does DevOps). High-level executives are not a part of our focus, as they are typically not actively involved in the development of software; but those who are in intermediary managerial positions where there is an active involvement with the development of software are suitable, provided they work in a DevOps manner. Some titles for the people who fit the latter description are: "Tech lead", "Lead Engineer", "Team Lead"; but they can also identify simply as "DevOps Engineer"s. These people in intermediary managerial positions are also suitable for our research since they are also actively engaged with the software development process, as such, they have the necessary technical knowledge associated with the DevOps capabilities. The DevOps Engineers who do not have managerial duties naturally also possess such knowledge. On the other hand, high-level executives (e.g. "C-level" executives) do not have such knowledge as they are typically not directly involved with the software development process, but rather focus more on the business-related duties.

The level of seniority is of no concern to the research, but the time the interviewees have spent working on their current function and the current organization is, since if the interviewees don't have enough experience within the organization/the function, they may have limited knowledge of the practices being followed in the organization and as such, their knowledge will not be of much use to our research. We have determined that for our candidates, both the experience within the current work function and the experience within the current company should be at least a year.

In addition to our main focus we are also targeting those who are not directly involved with the software development, but provide some additional services to teams within organizations who do, by the means of providing information and assistance related to DevOps/Agile methodologies and practices. Such people typically have titles such as "DevOps Coach", "DevOps Consultant", "Agile Coach" or "Agile Consultant". The people who work in such positions typically have some prior experience working in a DevOps setting and since their work typically involves trying to make teams operate in a way that is in line with the DevOps way of working, they can also provide some information on DevOps practices.

Since we expect the comments from "DevOps Engineer"s regarding their work to be more practical in nature, the main priority lies there, as our main purpose with the interviews (especially in the first round) is to learn more about the routines in practice. However, coaches/consultants often use maturity models or other means of assessment tools to understand about their teams/clients (whereas the "DevOps Engineers" are more or less the "subject" of such assessments and not the actual user), so their input on maturity models could be very useful and vastly different from engineers.

Chapter 5

Results

5.1 Interviewee Sample

Our sample for the first round of interviews consists of 10 people in total. 9 of these people identify as "DevOps Engineer"s, and 1 person identifies as a "DevOps Coach". We have reached out to approximately 160 people identifying as a "DevOps Engineer", and approximately 40 people identifying as a "DevOps Coach" using the means of personal networks and LinkedIn. This results in a 5.625% response rate for DevOps engineers and a 2.5% response rate for DevOps coaches (approx.).

Out of the nine DevOps engineers, two have extensive experience within their organization (15+ years), five have 2 to 10 years experience within their organization, two have 1 to 2 years of experience within their organization. The single DevOps coach in our sample has 1 to 2 years of experience working as a DevOps coach with 15+ years of prior experience working as a Software Engineer (but not as a DevOps engineer during all those years, as DevOps wasn't even around 15 years ago).

For the second round of interviews, the sample consisted of 11 people. 8 of these participants were participants from the last round. The 3 new participants all identify as a "DevOps Engineer". The single "DevOps Coach" was present in the second round as well, so this makes our sample 10 "DevOps Engineer"s and one "DevOps Coach". All 3 of the new participants were among the same 160 people we have reached out within the first round. The participants were unable to attend the first round due to personal matters, but were able to attend the second round. This boosted the response rate of DevOps engineers to 6.25% for the second round. Two of the new participants have 2 to 5 years experience within their organization, and one has 15+ years of experience. The two participants from the first round, who dropped out of the second round both had 15+ years of experience.

All the participants were working in organizations headquartered in Europe at the time of the interviews. The distribution among the European countries will not be given for to keep the anonymity of participants and their organizations. More details about the participants will also not be given for the same reason.

The overview of the sample is also represented in table 5.1

	First Round	Second Round
Number of Participants	10	11
Participant title (number of participants with title)	DevOps Engineer (9) DevOps Coach (1)	DevOps Engineer (10) DevOps Coach (1)
Participant experience in their current organization and function expressed in years (number of participants with experience)	15+ (3) 2-10 (5) 1-2 (2)	15+ (2) 2-10 (7) 1-2 (2)

Table 5.1: Overview of sample

5.2 Insights from first interviews

Following our first round of interviews we have gathered many valuable insights and observations. The insights are listed in this section in no particular order.

In total, 6 of the 10 participants commented about their experiences with maturity assessments; all 6 of these participants had experience being assessed with one, but one participant had experience as an assessor as well (this was the participant that identified as a DevOps coach). The remaining 4 out of the 10 participants did not have any experience with maturity assessments.

None of the participants had experience using a mainstream DevOps maturity model (e.g. one of the models featured in the works of Zarour et al. or a similar model) [9]. The mean of assessment for the participants was instead by answering questions typically provided by a "lead/manager" or an "agile/devops coach" present within the organization. The questions can be provided in the form of a filled questionnaire or by talking to people directly.

When it comes to being assessed, the shared consensus among 5 out of the 6 participants experienced with maturity assessments is that people are bothered when the assessment takes too long and/or is too complex, regardless of the mean of assessment. The remaining 1 participant did not comment on their feelings with their experience.

2 of the 6 participants had positive experiences with maturity assessments, and achieved various improvements following these assessments, while 4 of the participants commented that there weren't any meaningful improvements following the assessment. The 2 participants with positive experiences both noted that the assessments took place periodically (periods expressed in months) and the intent of assessment was self improvement (both on an individual level and on a team level). On the other hand, among those who had negative experiences with maturity assessments, 3 participants noted that the assessments did not take place periodically (i.e. assessment occurred only a single time), and one participant noted that the period of assessment was expressed in years (one and a half year to be exact). When it comes to the intent behind these assessments, among those 4 participants, 2 noted that they did it because they were asked to carry out the assessment as requested by their higher-ups, and 2 noted that they did it as per recommendation from someone within their network.

One of the most significant observations is that all 6 participants noted that doing a maturity assessment does not directly inform the users about what to improve to increase maturity. This confirms that the actionability problem we have highlighted within our problem statement is in fact a problem encountered in the practice. Furthermore, all 10 participants agree that improving/adopting the practices listed would help their organizations be more mature. This confirms that introducing capabilities in a model could be a practical solution to address the actionability problem.

When talking about self-empowerment within teams, 3 of the 10 participants noted that they may have "too much" autonomy, and 1 participant noted that they had a experienced a similar problem in the past. The participants talked about the various problems they faced regarding different areas. Such problems mentioned are related to management and the culture within the organization (the Lean and Culture dimensions of CALMS). More specifically, indecisiveness regarding decision-making, and non-standardized and unregulated use of tools across different teams are the problems mentioned by the participants. One participant also talked about more problems such as different teams having drastically different approaches to deployment and testing, and even with their way of working (agile vs waterfall); which are not just related to the Lean and Culture dimensions but also the Automation dimension of CALMS.

Participants were also asked to self-assess their own maturity, without the use of a model, and speculatively, in order to get insights as to how they view themselves when it comes to maturity. 3 participants noted that it's difficult to be completely mature regarding any practice/dimension at any given time, as what's considered mature is constantly changing.

When we consider the participants' comments about their practices and try to hypothetically assess their maturity using a maturity model (i.e. thinking about where would they place within a maturity model based on their comments), it is difficult to assess their level of maturity in any dimension, since their descriptions often make it such that they are not at an exact level but rather somewhere between two distinct levels.

One participant had extensive knowledge about tools, and could talk about their own tools elaborately, and knew which tools on the market could benefit them in what kind of manner, this may indicate proficiency related to certain technical practices. On the other hand, another one of the participants admitted to using tools and adapting practices that "came along with the tools", which may indicate they are not as proficient when it comes to the same practices. Therefore, approach to tools may also be relevant to maturity.

As shown in the interview guide, one of our discussion points was about the evolution, the adoption and the order of importance related to the practices. All participants comments indicate that all the practices, even the ones featured across different CALMS dimensions, are related to one another. This is not surprising, however such relations were not the primary emphasis of the interviews, as such, more exploration to these relations will be needed.

Furthermore, the participants' remarks on practices indicate that all the practices are meant to be adopted and evolved together. The concept of "continuous improvement" was explicitly mentioned by five of the participants during this part of the interviews, and the shared belief between the participants was that as long as there was "continuous improvement" on all fronts (i.e. with all CALMS dimensions), then their organization was on the right direction.

That being said, an important thing to note was that there were differences between some participants' choices as to what practices are considered more important than others regarding adoption/improvement of practices. For example, one of the participants put the highest importance on the practices under the monitoring dimension of CALMS (especially the centralized logging practice), while another put the highest importance on the practices under the automation dimension (especially practices that are related to increasing the reliability of their deployment processes such as automated deployment); even though both of these participants were among the aforementioned five participants who explicitly talked about "continuous improvement". That's not to say that these participants only care about these particular dimensions and not the others, it only shows that a difference in priorities can exist. The reason behind why these differences are in place will be discussed in the next chapter.

5.3 Improved Maturity Model

After completing the first round of interviews, we took an existing DevOps maturity model and improved it based on our findings and the literature. In this section, we will describe the model, how to use it, and the design choices behind the model and how we have came to create it.

5.3.1 Description of the model

Our model with all its pages is featured in Appendix B. On the first page that the user sees, we feature the maturity model of Eficode without any changes (see 2.1). As explained earlier in the literature chapter, Eficode's model features 6 dimensions ("Organization and Culture", "Environments and Release", "Builds and Continuous Integration", "Quality Assurance", "Visibility and Reporting", "Technologies and Architecture"), and 4 levels (numbered from 1 to 4, 1 being the lowest and 4 the highest). For each dimension/level pair, a broad description of how related activities are supposed to look like is given, and users are meant to pick the description that best fits them for each dimension, which results in their assessed level of maturity across 6 different dimensions. In addition to the 6 listed dimensions, there is also one unlabeled dimension at the top. This dimension is primarily on the difficulty regarding starting new development projects. For future reference this dimension will be called "Starting new projects", and all the other dimension except this one will be called "the main dimensions".

For each of the main dimensions, a button is placed on their labels such that when the user clicks them, they move to a different page featuring a list of capabilities relating to the dimension. Such capabilities are listed under 4 different "capability categories": "Cultural capabilities" (corresponding to "Organization and Culture" dimension, figure 5.1), "Continuous Delivery capabilities" (corresponding to all three dimensions of "Environments and Release", "Builds and Continuous Integration", and "Quality Assurance", figure B.9 in the Appendix), "Lean Management and Monitoring capabilities" (corresponding to "Visibility and Reporting", figure B.24 in the Appendix), and "Architectural capabilities" (corresponding to "Technologies and Architecture" dimension, figure B.20 in the Appendix).

For example, when the user clicks on the "Organization and Culture" dimension, they will move to the page featuring the Cultural capabilities, as shown in figure 5.1.

In each of these lists of capabilities, various number of capabilities are featured. The capabilities associated with each capability category is given in table 5.2. The user can click on the arrow symbol next to a capability (see figure 5.1) to move to a different page, featuring more in-depth information about the capability.

For example, if the user clicks on the arrow symbol next to the "Fostering and enabling team experimentation" they will move to a page featuring more information related to this capability, as shown in figure 5.2.

Within the capability specific pages, information is provided under 3 separate blocks that are referred as "sections". Under the first section, a description of the capability and the typical work that entails it is given. This section is titled either "What is [x]?", where x is the name of the capability, or "What is this about?"; with the exception to "Encouraging and supporting learning" capability, where the title is "Approach to learning". This section is also referred as the "what" section.

Proceeding with the previous example of "Fostering and enabling team experimentation", in this section, what the work related to team experimentation entails is described (see figure 5.2).

Under the second section, reasons as to why the organizations should have this capability is given. Such reasoning is all taken from literature (more information on these literature is given in section 5.3.3), but for the sake of visuals within the model, these are not cited within the

Category	**Capability**
Cultural	Supporting a generative culture
Cultural	Encouraging and supporting learning
Cultural	Facilitating collaboration among teams
Cultural	Increasing job satisfaction
Cultural	Embodying transformational leadership
Cultural	Fostering and enabling team experimentation
Continuous Delivery	Using version control for all production artifacts
Continuous Delivery	Automated deployment
Continuous Delivery	Implementing Continuous Integration
Continuous Delivery	Trunk-based development
Continuous Delivery	Implementing Test Automation
Continuous Delivery	Supporting test data management
Continuous Delivery	"Shifting left" on security
Continuous Delivery	Implementing Continuous Delivery
Continuous Delivery	Gathering and implementing customer feedback
Continuous Delivery	Working in small batches
Architecture	Loosely coupled architecture
Architecture	Architecting for empowered teams
Lean Mgmt and Monitoring	Lightweight change approval processes
Lean Mgmt and Monitoring	Monitoring across application and infrastructure to inform business decisions
Lean Mgmt and Monitoring	Proactive system health checks
Lean Mgmt and Monitoring	Using Work-in-process (WIP) limits to manage work
Lean Mgmt and Monitoring	Visualizing work to monitor quality and communicate within the team
Lean Mgmt and Monitoring	Making the flow of work visible through the value stream

Table 5.2: Capabilities featured in the model

Figure 5.1: List of Cultural capabilities

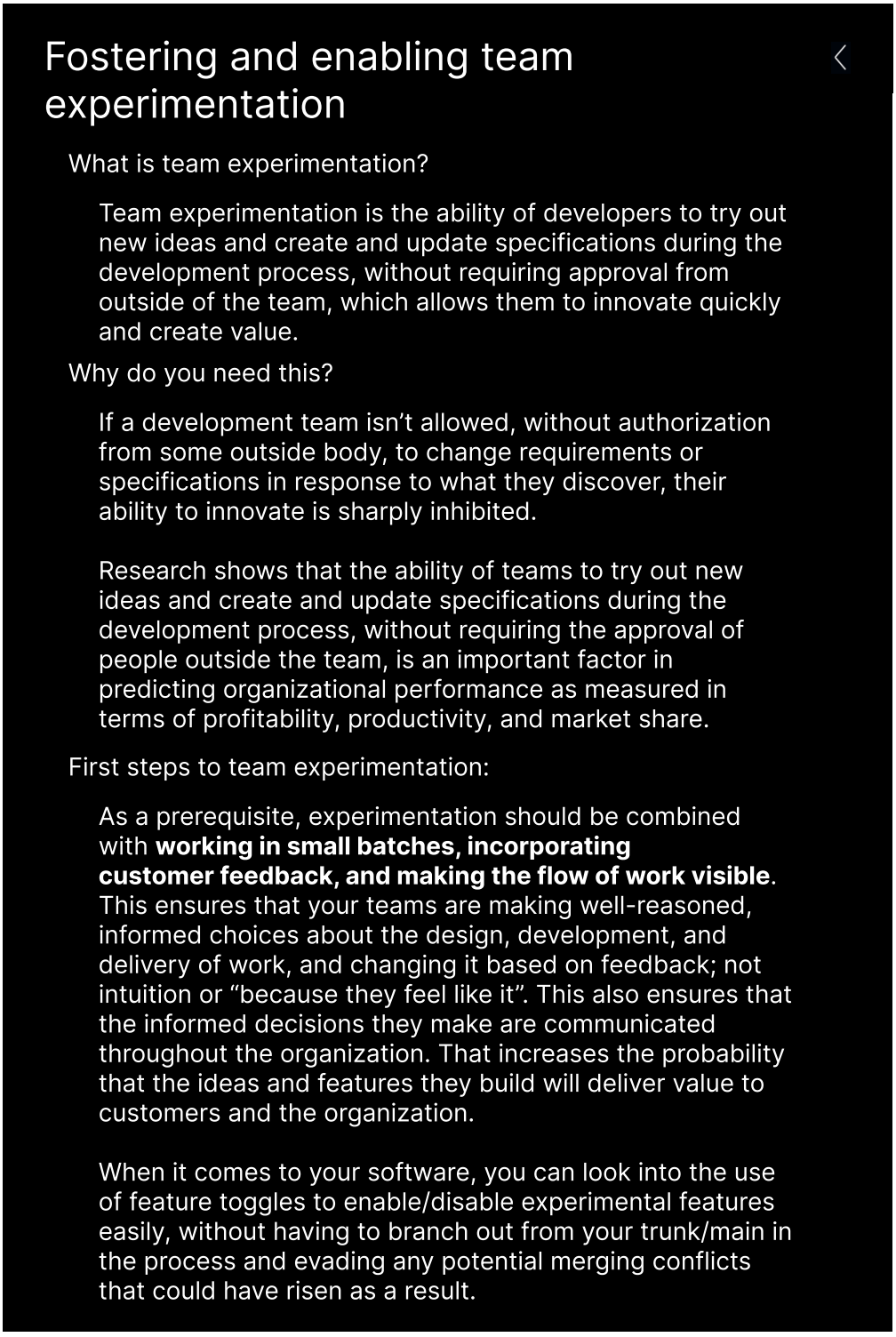

Figure 5.2: Capability specific page of the "Fostering and enabling team experimentation"

text. The title for this section is "Why do you need this?", and the section is also referred as the "why" section.

For example, for "Fostering and enabling team experimentation", team experimentation's relation to innovation, and how it impacts organizational performance is mentioned as to why organizations should establish this capability (see figure 5.2).

The third and the last section describes the first steps to adopting the capability in question, how it should be like in the long run, the best practices regarding the capability and common pitfalls. The title for this section is "First step to [x]", where x is the name of the capability; and the section is also referred as the "first steps" section.

with the "Fostering and enabling team experimentation" example, the suggestions given are to establish some of the related capabilities in order to ensure the experiments are based on well-reasoned choices rather than intuition, and the use of feature toggles with software development to quickly enable and disable experimental features (see figure 5.2).

The "what" section is not present in the following capabilities: "Facilitating collaboration among teams", "Increasing job satisfaction", "Using version control for all production artifacts", "Automated deployment", "Supporting test data management", "Gathering and implementing customer feedback", "Working in small batches", "Architecting for empowered teams", "Visualizing work to monitor quality and communicate within the team", "Making the flow of work visible through the value stream". The reasoning for this is given in section 5.3.3.

In some of the pages containing detailed information about a specific capability, under the "why" section and/or the "first steps" section; there are references to specific capabilities and/or capability categories, highlighted in bold text. These texts are also clickable by the user, and doing so will lead the user to the page related to the specific capability or the capability category that was clicked on.

As seen in figure 5.2, within the "first steps" section in the page of "Fostering and enabling team experimentation", there are links to the pages of "Working in small bathes", "Gathering and implementing customer feedback", and "Making the flow of work visible through the value stream" capabilities.

There are also two additional pages specific to two different capabilities. In the page related to the "Loosely coupled architecture" capability and the page related to the "Making the flow of work visible through the value stream" capability, there are certain pieces of text highlighted in yellow. Clicking on these leads the user to a page featuring further information to the highlighted topic, along with provided visuals (figures B.22, and B.31 in the Appendix).

5.3.2 How to use the model

As described earlier, with Eficode's model, the users are expected to assess themselves by picking the description that best fits them for each dimension, which results in their assessed level of maturity across various dimensions. As we've took this model as our basis, this fundamental use still holds true; the users are still expected to assess themselves for each dimension. However, the capabilities that we have introduced provides more information regarding each dimension, as such the users will be able to make more accurate assessments.

We believe that the best way to use the model is as follows: First the user is expected to set their sights to one of the main dimensions (any of the 6 labelled dimensions), then click on the dimension name to view the list of capabilities regarding that dimension. The users are then expected to view the capabilities, and get further information on any of the capabilities that strike them the most. We encourage the users to primarily focus on capabilities that they do not already possess, or the capabilities that they have plans for adopting/improving; as the information provided within the "first steps" sections can be very beneficial for establishing these

capabilities. The users can of course get information on the capabilities that they possess as well, and they can benefit from doing so by comparing the current state of their capability with what's described, or seeing the best practices and the ideal long term state regarding the capability.

After getting capability specific information regarding a dimension, the users are expected to go back to the first page (featuring the original model of Eficode, figure B.1), and (in order of importance), based on the information regarding the capabilities that they have viewed, and based on the basic information provided within each level, the users should give themselves a rating between 1 and 4 for that particular dimension. When giving themselves a rating, if the user feels torn between two levels, or feels close to a certain level but not quite there yet, they can give themselves a fractional rating as well (e.g. there is no strict requirement to be a level 2 or level 3 if the user feels somewhere in between, and they can simply be a level "2.5"; or if they feel close to level 3 but not exactly three, they can be level "3.9"). This is in place because we have found from our first round of interviews that the descriptions of the users on their practices often do not perfectly match with the level descriptions given in one level or another; doing assessments this way also makes them more "dynamic" (more information will be provided on this in the next section).

The process then should repeat until the users gives themselves a rating for each of the main dimensions.

Afterwards, the user should look at the unlabelled dimension ("starting new projects" dimension) and give themselves a rating between level 1 and level 4, again with the option to have a fractional rating. The rating that the user gives themselves regarding this dimension should be representative of the ratings related to the main dimensions (i.e. the rating for the "starting new projects" dimension should be within 1 point range of the average rating across the main dimensions). If that's not the case, then either the user have not rated themselves honestly regarding one or multiple dimensions; or it can't be claimed that the organization that the user belongs in is doing DevOps. This marks the end of a single use of the model.

That being said, it is important to note that the model should not be used only once. One of the key insights from our first round of interviews was that the people who have achieved improvements following maturity assessments (i.e. the "successful practitioners"), did two things different compared to the people who did not see such improvements (i.e. the "unsuccessful practitioners"). Firstly, the successful practitioners have not assessed only a single time, but multiple times, and periodically. Each organization's rate of development is different, therefore it's hard to set exact limits on such periods; but our findings indicate that the period of these assessments should at least be expressed in months and not years, and it should stay at a relatively consistent pace (i.e. assessments are to be scheduled and performed on time, and not to be performed only when teams feel ready for it). Secondly, the intent behind the assessments with the successful practitioners was self-improvement, both on an individual level and on a team level; while unsuccessful practitioners did such assessments either because they were ordered to do so by their higher-ups, or because they were influenced by someone within a different organization to do so. So each time the users use the model, their intent should be on self-improvement. Organizations can have different views on what they see as improvement; but as a suggestion, the users can try to identify certain capabilities to improve upon and/or adopt capabilities not currently present. They can do so by focusing on the dimensions they score the lowest, and looking at the capabilities related to that particular dimension; though, there isn't a strict requirement to focus on the dimension they score the lowest, and they can choose to focus on other dimensions if they have good reasons for doing so. In any case, the next time the users use the model, they should see whether they are on the right track with the capabilities they have chosen to improve/adopt, and whether they score higher than the last time they have used the model, on dimensions related to these capabilities.

5.3.3 Designing the model

We have started our journey to design the model by selecting a base model to improve upon. Two candidates stood out from the rest, one of them was Eficode's model, the other was the model of Feijter et al. [30] [33]. As mentioned in earlier chapters, Eficode's model was already supported as a useful model by Zarour and colleagues, and was prevalent within other literature (including the work of Offerman et al., which we have utilized for the first round of interviews) [9], [26]. As mentioned earlier, the model of Feijter et al. was the only DevOps maturity model we have found to feature capabilities, but it had its flaws (see chapter 4 for more information). Furthermore, we have thought that the complexity of the model could be a problem but we weren't sure how big of a problem that would be for practitioners. One of the key findings from our first round of interviews showed that all the participants experienced with maturity assessments indicated that they were bothered when the assessment procedure was too complicated; showing that this complexity problem was more important than we initially thought. This pushed us to use Eficode's model instead of the model of Feijter et al. as our base model to improve upon.

After making the decision to go through with Eficode's model, we have decided to finalize the list of capabilities to feature in the model. We have previously used the practices and the capabilities featured in the works of Plant et al. and Offerman et al. for our first round of the interviews. We have classified these practices and capabilities under 4 main categories, with respect to the CALMS framework. The categories are "Culture and Sharing", "Automation", "Monitoring", and "Agile/Lean". We have combined the "Culture" and the "Sharing" categories together as the similarities of practices shared between the two made it reasonable to do so; and we have added "Agile" to the "Lean" for the same reason.

As a result of the first round of interviews, we have gathered some valuable data on these capabilities and practices. Then, using the aforementioned 4 categories, we have created some codings on our interview transcripts. You can find all of these codings under their respective categories in figures 5.3, 5.4, 5.5, 5.6. The unique discussions related to each coding is counted and the frequencies are shown in figures 5.7, 5.8, 5.9, 5.10.

After we had our data available, we tried to came up with a solution that was helpful to solve the problems we have tried to address, was easy to implement within the Eficode's model seamlessly, and also was easy to use. Unable to come up with such a solution, we have gone back into the literature to help us.

On this journey, we have dived deeper into the "Accelerate" book which we have looked into before [3]. As explained earlier within the literature chapter, Accelerate talked about DevOps capabilities, and was also co-authored by the primary creator of the CALMS framework. In the book, there are 24 capabilities, listed under 5 categories: "Cultural", "Continuous Delivery", "Architecture", "Product and Process", and "Lean Management and Monitoring". In the book, when describing the capabilities featured, the focus is on the best practices and routines related to these capabilities, why such practices and routines are better, and how can organizations implement them; so the explanations are geared towards practical use. Whereas within the works of Plant et al. and Offerman et al., the capabilities and the practices are described in a way that is capturing all that is done related to these capabilities and practices, which provided a broad perspective of these capabilities and practices [2], [26]. This broad perspective turned out to be great to set a basis for discussion for our first round of interviews, since it allowed more room for participants to relate to certain capabilities and practices, while not leading them towards certain directions that could result in bias. On the other hand, they weren't very useful when it came to improving Eficode's model. This practical approach within the Accelerate book was exactly what we could use. Knowing Eficode's model was made with respect to CALMS framework was another reason to use the capabilities from Accelerate [26]. So, we tried to came

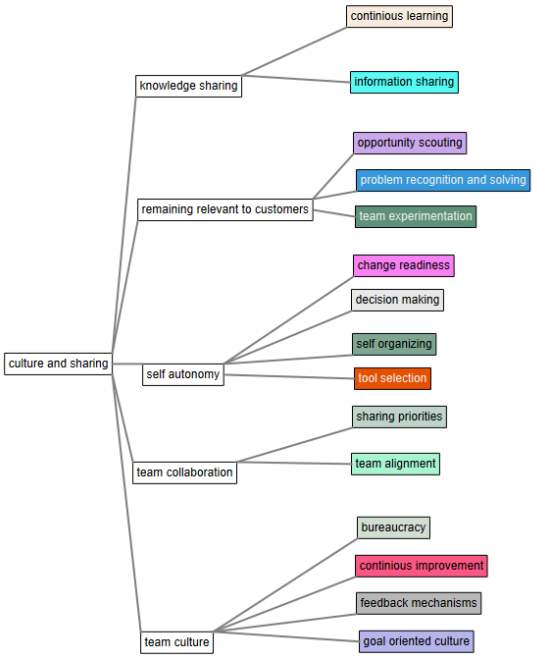

Figure 5.3: The codings related to culture and sharing

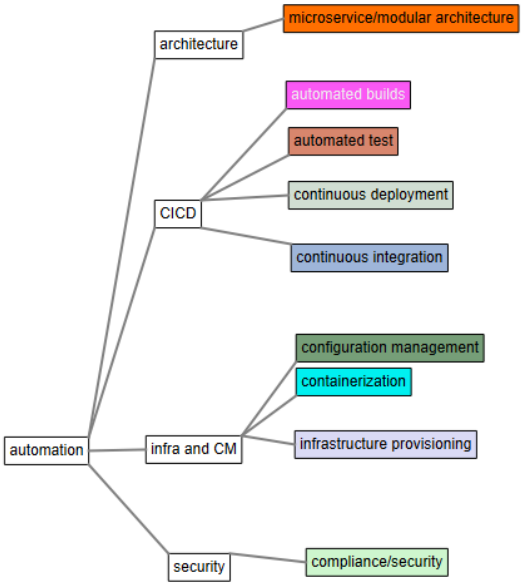

Figure 5.4: The codings related to automation

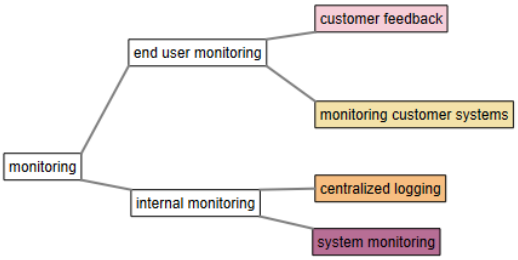

Figure 5.5: The codings related to monitoring

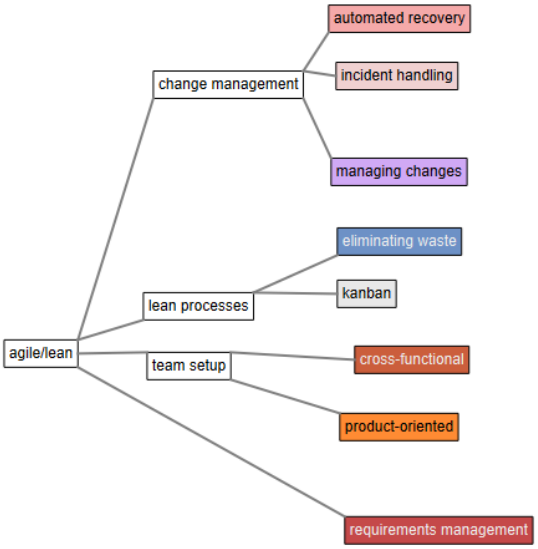

Figure 5.6: The codings related to agile/lean

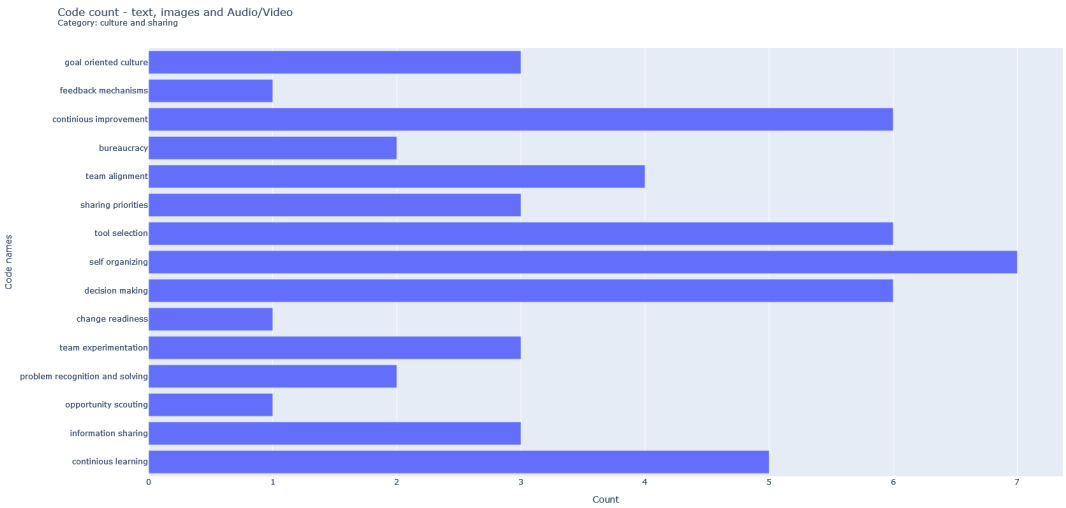

Figure 5.7: The frequency of codings related to culture and sharing

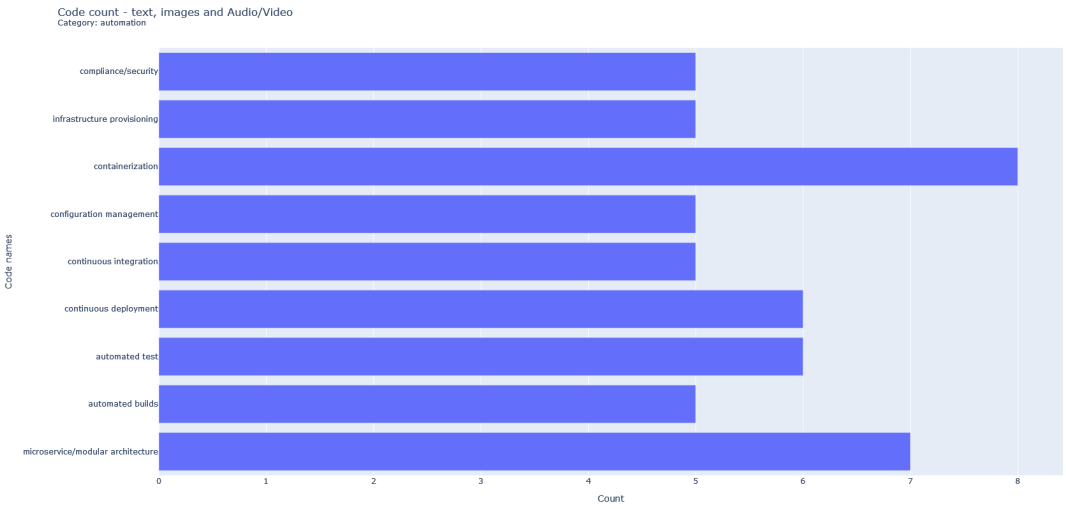

Figure 5.8: The frequency of codings related to automation

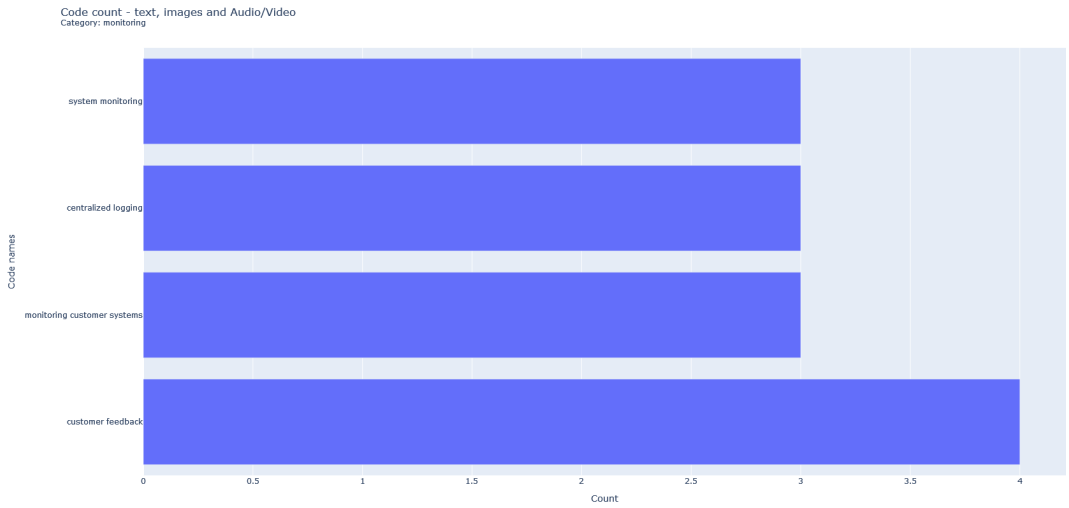

Figure 5.9: The frequency of codings related to monitoring

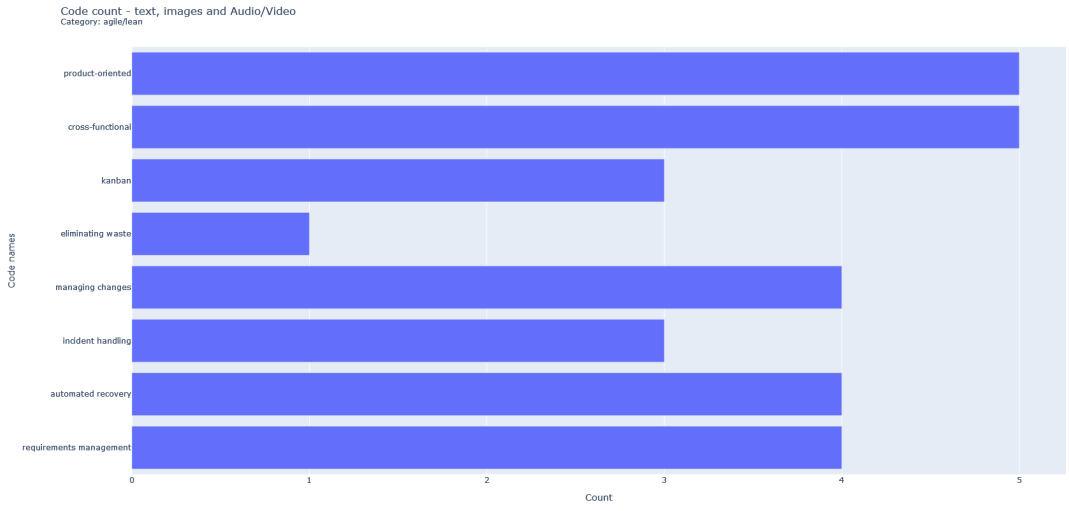

Figure 5.10: The frequency of codings related to agile/lean

Eficode Dimension
Organization and Culture
Environments and Release
Builds and Continuous Integration
Quality Assurance
Visibility and Reporting
Technologies and Architecture

Table 5.3: Eficode dimensions and their assigned colors for mapping capabilities

up with a solution to feature the capabilities given in the Accelerate book within the model.

For such a solution, we have started by mapping the capabilities featured in Accelerate with the dimensions of Eficode's model (see table 5.3 and 5.4). As you can see with these mappings, mapping to "Organization and Culture", "Visibility and Reporting", and the "Technologies and Architecture" dimensions were straightforward, as the capabilities for these were all under their respective capability categories ("Cultural", "Lean management and monitoring", "Architecture"). However with "Continuous Delivery capabilities" and the "Product and Process capabilities" a solution wasn't so obvious.

With the "Continuous Delivery capabilities", we have decided to group the "Builds and Continuous Integration", "Environments and Release", and the "Quality Assurance" dimensions together and map the capabilities to all three at once, instead of separating them; and assigning them to the "Continuous Delivery capabilities". These capabilities are highly related to one another anyway (you can observe such connections within specific capability pages featured in the Appendix), further making this grouping decision sensible.

With the "Product and Process capabilities", it was apparent that all 4 capabilities were related to a different Eficode dimension, so, we have decided to map each of these specific capabilities to the dimension they related to. This solution seemed like the most straightforward choice, as the alternatives would have been either having to group them together with a different category of capabilities, which wasn't ideal because none of the capabilities heavily related to another category of capabilities; or we would need to create a new dimension within the Eficode model, which wasn't ideal either because it would mean directly altering the foundational model of our choice that we have selected based on literature.

After mapping the capabilities, we had to decide on the information that we would feature within the pages of each specific capability. We had already decided to use Accelerate book as our primary source for such information, but we have also consulted the "DevOps Handbook" at times where we needed more in-depth information regarding capabilities [3] [28]. Furthermore, we were aware that we needed to feature the information in a condensed manner to avoid it making the model complex. This led us the idea to categorize our information and limit it to certain and specific sections:

The "first steps" section was rather trivial to come up with, as it was necessary to feature the practical information that would benefit the users. Such information was also abundant within the Accelerate book, so all the practical information in all capabilities are coming from the Accelerate book. As mentioned earlier, these information describes the first steps needed to adopt a particular capability, what the users can do in the longer term related to this capability, the best practices, common pitfalls and links to related capabilities (if any exists).

There was also a need to explain the rationale behind why the users should adopt these capabilities in the first place, which resulted in the creation of the "why" section. For many of the capabilities, the Accelerate book gives the scientific reasoning as to why these capabilities are

Accelerate Category	Capability
Cultural	Supporting a generative culture
Cultural	Encouraging and supporting learning
Cultural	Facilitating collaboration among teams
Cultural	Increasing job satisfaction
Cultural	Embodying transformational leadership
Continuous Delivery	Using version control for all production artifacts
Continuous Delivery	Automated deployment
Continuous Delivery	Implementing Continuous Integration
Continuous Delivery	Trunk-based development
Continuous Delivery	Implementing Test Automation
Continuous Delivery	Supporting test data management
Continuous Delivery	"Shifting left" on security
Continuous Delivery	Implementing Continuous Delivery
Architecture	Loosely coupled architecture
Architecture	Architecting for empowered teams
Product and Process	Gathering and implementing customer feedback
Product and Process	Making the flow of work visible through the value stream
Product and Process	Working in small batches
Product and Process	Fostering and enabling team experimentation
Lean Mgmt and Monitoring	Lightweight change approval processes
Lean Mgmt and Monitoring	Monitoring across application and infrastructure to inform business decisions
Lean Mgmt and Monitoring	Proactive system health checks
Lean Mgmt and Monitoring	Using Work-in-process (WIP) limits to manage work
Lean Mgmt and Monitoring	Visualizing work to monitor quality and communicate within the team

Table 5.4: Capabilities featured in the Accelerate book, along with its mapping to Eficode dimensions

Capability	Related Codings
Supporting a generative culture	Goal-oriented culture, Continuous Improvement, Bureaucracy
Encouraging and supporting learning	Continuous Learning, Information sharing, Opportunity Scouting, Team experimentation, Continuous Improvement
Facilitating collaboration among teams	Sharing Priorities, Team alignment, Problem recognition and solving, Decision making
Increasing job satisfaction	Self-organizing, Tool selection
Embodying transformational leadership	Continuous Improvement, Continuous Learning, Information sharing, Opportunity Scouting, Problem recognition and solving, Change readiness
Fostering and enabling team experimentation	Team experimentation, Requirements management, Opportunity Scouting, Change readiness, Decision making

Table 5.5: Mapping of codings to "Cultural" Capabilities

the better solutions, so the information on these are also primarily coming from the Accelerate book, and secondarily from the DevOps Handbook.

In addition to this, we have anticipated that some of the capabilities may not be well understood or known by the general audience, as such, the "what" section was created to inform any potential users about what a specific capability entails. The information given in these are also taken primarily using the Accelerate, and secondarily the DevOps Handbook; but the information coming from the latter primarily lies within these sections. The "what" section is not present in some of the capabilities as mentioned earlier. The reason for this is because such capabilities are either self-explanatory by their titles alone, or the information given under the other two sections (the "why" section and the "first steps" section) makes it trivial what the capability is about. The exact list of capabilities not to feature this section is given in section 5.3.1.

The information provided within the capability specific pages were also done with respect to the insights on the practices and capabilities gathered from the first round of interviews. Tables 5.5, 5.6, 5.7, 5.8 shows the mapping between all the capabilities featured in the model and the codings that correspond to the various capabilities and practices discussed in the first round of interviews.

One insight from our first round of interviews was that there were significant connections between the practices that the participants talked about, but we were unable to make meaningful relations between such connections as we did not focus on those at the time. As we have dived deeper into our insights and the literature; and filled the capabilities with more and more information as a result, the connections between the capabilities also became more and more apparent. As such, we have decided to feature links between capabilities that relate to each other, and highlighted such links with texts written in bold. We have established these links based on literature and our findings from the first round of interviews. You can observe the links between the capabilities within the specific capability pages featured in the Appendix; and also within the tables mapping the codings and capabilities.

Another thing that became apparent was the need to feature visual aid within certain capabilities, but we did not want to put such visuals directly within the capability specific pages, as it would take a lot of space and reduce ease of use. As such, we have decided to highlight

Capability	Related Codings
Using version control for all production artifacts	Configuration management, Infrastructure provisioning, Automated builds
Automated deployment	Automated builds, Automated recovery, Continuous Deployment, Containerization
Implementing Continuous Integration	Automated builds, Automated test, Continuous Integration
Trunk-based development	Continuous Deployment
Implementing Test Automation	Automated test, Feedback mechanisms
Supporting test data management	Automated test
"Shifting left" on security	Compliance/security, Feedback mechanisms, Automated Test
Implementing Continuous Delivery	Automated builds, Automated Test, Continuous Deployment, Continuous Integration, Continuous Improvement, Feedback mechanisms
Gathering and implementing customer feedback	Customer feedback, Continuous Integration, Product-oriented team setup
Working in small batches	Continuous Deployment, Continuous Improvement, Product-oriented team setup, Continuous Learning, Feedback mechanisms

Table 5.6: Mapping of codings to "Continuous Delivery" Capabilities

Capability	Related Codings
Loosely coupled architecture	Microservice/modular architecture, Infrastructure provisioning, Cross-functional team setup, Change readiness,
Architecting for empowered teams	Self-organizing, Compliance/security, Tool selection, Product-oriented team setup

Table 5.7: Mapping of codings to "Architectural" Capabilities

Capability	Related Codings
Lightweight change approval processes	Incident Handling, Managing changes, Change readiness, Compliance/security, Self-organizing, Bureaucracy
Monitoring across application and infrastructure to inform business decisions	Monitoring customer systems, Centralized logging, System monitoring, Problem recognition and solving, Decision making,
Proactive system health checks	System monitoring, Problem recognition and solving
Using Work-in-process (WIP) limits to manage work	Kanban, Eliminating waste, Feedback mechanisms
Visualizing work to monitor quality and communicate within the team	Kanban, Feedback mechanisms, Sharing priorities
Making the flow of work visible through the value stream	Continuous Improvement, Sharing Priorities, Team alignment, Product-oriented team setup, Cross-functional team setup, Goal-oriented culture, Requirements management

Table 5.8: Mapping of codings to "Lean Mgmgt and Monitoring" Capabilities

certain parts of the text with a different color and add links to them, connecting them with a different page that feature the visuals that we wanted the users to see (i.e. the "Loosely coupled architecture" capability and the "Making the flow of work visible through the value stream" capability, as described in section 5.3.1. See Figures B.22, and B.31).

As mentioned earlier, making the model simple was an important consideration. Eficode's model was quite simple, but obviously, introducing capabilities to the model were expected to increase its complexity; so, we tried to make the model as simple as we can, the most significant design choice in this regard is giving the users the freedom to view the capabilities that they choose, instead of requiring them to check every capability.

Making the model simple was not our only requirement though, as it needed to address the problems we were trying to solve as well. The actionability problem was one of the key issues that we were trying to address with our improved model, and our first round of interviews proved it to be a real problem encountered in practice. The capabilities we have introduced to the model act as the solution to this problem. The capabilities provide additional layers of information to make these assessments more accurate and easier for the users. The capability specific information enables the users to see how they can improve regarding a specific capability, or adopt in the first place. As a result, this well help the users improve their maturity.

The second problem we needed to address was that our model needed to utilize dynamic capabilities. The works of Plant et al. showed us that "it is possible to define a specific set of capabilities that are relevant to DevOps teams but that any measurement instrument of capabilities will need to capture various configurations of the same capability in order to account for their idiosyncratic implementation"; which meant that in order to feature capabilities, we had to capture "numerous possible configurations of a capability instead of merely assessing whether a capability is performed at a sufficient level or not" [2]. This is in fact in line with what we have done within our "first steps" sections, as the information about the ideal state of the capabilities, both in the short and the longer terms, the best practices and the common pitfalls serve a foundation of information that cover numerous possible configurations for a capability. The users being able to rate themselves fractionally instead of having to adhere to strict levels also relates to this; as doing so enables the users to identify themselves in a wider spectrum according to the various states of their capabilities, hence more configurations of capabilities are covered.

5.4 Insights from second interviews

After we had finalized our maturity model, we have started the second round of interview to validate it. Following are the insights we have gathered as a result of these interviews:

One of the aspects we have checked for validation within our model was how well the participants could relate to the information provided in the model, especially the information that was given within the capability specific pages. Out of the 11 participants, 6 participants explicitly said that the suggestions given within the capability specific pages (in the "first steps" sections) were exactly in line with their own personal experiences. While the remaining 5 participants did not made such a statement explicitly, they could all relate to the information given in multiple specific capability pages that we have covered during our assessment and found the suggestions sensible.

6 out of the 11 participants explicitly noted that the suggestions given in the specific capability pages were helpful to improve themselves regarding those particular capabilities, and that implementing those suggestions would lead them to an increased maturity (and as a result would lead to higher scores on the model). The remaining 5 participants did not explicitly state that

because they have found such suggestions rather trivial, but that still showed that they agreed with these suggestions.

5 out of the 11 participants noted that there was a mismatch between the information that was provided with the basic level descriptions regarding a particular dimension on the first page of the model (i.e. Eficode's original model), and the capabilities that corresponded to that dimension. This problem applied to all dimensions, but it was especially prevalent with the Visibility and Reporting dimension and the Lean Management and Monitoring capabilities that corresponded to it, as the participants noted that the basic level descriptions did not mention anything related to Lean Management at all; and regarding monitoring, the descriptions did not capture all the information related to the associated capabilities.

3 participants commented on the fact that being able to rate themselves fractionally made the assessment easier for them.

2 participants differed from the others in the way that they were a part of a DevOps team, but the organization had also separate development and operations teams as well. This way of doing DevOps isn't inherently wrong but it's different than the "traditional" DevOps described earlier within the literature chapter. In this setting, the DevOps team act as a bridge between the development and operations teams (which are 2 separate teams apart from the DevOps team and each other). The DevOps team's work still entails development and operations work to support those teams just like in a traditional setting, but also they handle most of the unplanned work that is not directly linked with operations, and problems that are inherently complex to solve, in order to reduce the workload on development and operations. These participants found that the model didn't capture their needs fully. One participant noted that for their case, centralized logging and alerting based on metrics (which were both mentioned within the "Monitoring across infrastructure and application to inform business decisions" capability) is "as important as Continuous Delivery" and that these two practices could have been separate capabilities on their own. The other participant noted that they did not think about the business side at all and they did not find related suggestions helpful for them (e.g. doing monitoring across different layers, including business, as described in the "Monitoring across infrastructure and application to inform business decisions" capability); since, in their case, the business requirements are handled by the development teams and they received only technical requirements.

The participant that had the highest maturity among all have scored at least a 3 on all dimensions, and they were the only participant to score a 4 on the "Builds and Continuous Integration", and the "Environments and Release" dimensions. This participant was able to identify strongly with nearly all of the capabilities and was among the 6 participants that found the suggestions helpful and in-line with their own experiences. This participant also brought up a lot of interesting ideas regarding the "Facilitating collaboration among the teams" capability (under "Organization and Culture" dimension) that we didn't see before in our sources. One of the suggestions given within this capability is to encourage people in the organization to move between teams (see figure B.5). The participant argued that the teams within an organization create a sub-culture over time, and when the sub-culture is "better" than the overall culture of the organization, that creates a comfort zone for the individuals in the teams, and such individuals don't like to make such moves and move outside of their comfort zone. They also said that if those individuals do indeed move to different teams with a "worse" sub-culture and succeed to establish the better sub-culture from their previous team there, that's very beneficial for the organization, but there is no guarantee that those individuals would succeed, and that they are more likely to end up becoming part of that worse sub-culture. This participant also noted that doing a systematic review of the state of their work made them think about the "big picture", which is something they don't get to do often as they get caught up in their work.

The only participant to identify as a DevOps coach, which in turn had practical experience

using maturity models, commented that the model provided a lot of information in a dense manner, which they personally liked, as they noted that most practitioners don't want to read literature extensively.

There were also some miscellaneous comments on the user experience and user interface. One participant said that it would be better if the "starting new projects" dimension had a label. Another participant noted that they would have liked to see a "connection map" of sorts, showing all the capabilities and the connections between the capabilities that have a link among each other (the connections which are indicated by bold text in capability specific pages). Another participant said that there could be a checkbox/point system implemented within the capability specific pages and that the users could have a score with the capabilities themselves too, which they believed would make the experience more concrete.

Chapter 6

Discussion

In this chapter we will discuss about out the insights we have gathered from our interviews, and the limitations of our research.

6.1 Implications of the insights on the model

6.1.1 Problems addressed by the model

One of the insights from our first round of interviews was that a lot of practitioners didn't like to be a part of assessments that were rather complex. As described in section 5.3, we have tried to make our model as simple as possible while trying to address the problems we have identified. As shown from the insights from the second round of interviews, all the participants could relate to the information given in the model. Furthermore, the comments of the DevOps coach who had practical experience using models to make maturity assessments, made remarks on how the model provided information in a compact manner. Considering these insights, it can be claimed that our model was validated as "easy to use".

As presented initially within our problem statement and mentioned throughout the paper multiple times, the "actionability problem" was one of the primary problems we have tried to address with our model. As mentioned earlier, the second round of interviews showed us that 6 out of 11 participants found the suggestions given in the model helpful, and agreed that implementing those solutions would indeed help them improve their maturity (and as a result these improvements would reflect on the model with increased scores). As a specific example, one of the insights from our first round was that multiple participants had problems with "too much" autonomy, we tried to address this problem by featuring information on how the degree of freedom should be handled within related capabilities ("Increasing job satisfaction" and "Architecting for empowered teams" are such capabilities, see figures B.6 and B.23). In the second round, the same participants suffering from this problem has found the suggestions featured helpful. Therefore, it can be claimed that our model was moderately successful in addressing the actionability problem, but it could use further improvements. More specifically, if we recall that the remaining 5 out of 11 participants found the suggestions quite trivial, and that the highest scoring participants' comments has revealed a perspective to one of the capabilities that we did not consider before, it could be argued that the information given in the model is not complete for all capabilities, and could benefit from a refinement that would make it more extensive. This would of course, require further literature and/or practical research. Furthermore, it is important to note that doing so would highly likely be at the expense of some increased complexity.

The other main problem we tried to address was featuring dynamic capabilities within our maturity model. As we tried to explain in this paper earlier, there isn't a list of dynamic capabilities that we can feature in our model and be done with it (i.e. they are dynamic). As mentioned in section 5.3.3, when designing our model, we tried to capture "numerous possible configurations of a capability" for every capability that we featured, which we believed would make our model "dynamic" as a result [2]. All the participants in the second round being able to easily relate to the information provided and discuss about certain capabilities, despite the participants having various differences (with regards to industry, technologies and tools used, assessed maturity, individual seniority, years of experience within the organization, and more information that wasn't available to us. We could not disclose all of these information for the sake of anonymity of our participants), do lead us to believe that our model can be claimed as one that features dynamic capabilities. Furthermore, we have previously argued that users being able to rate themselves fractionally was also related to this problem. If we recall that the second round of interviews showed that 3 participants explicitly stating that they liked being able to fractionally rate themselves, this makes our claim stronger.

As mentioned earlier, the participant who have scored the highest noted that using our model made them think about the "big picture", and said that it isn't something that they don't get to do often as they get caught up in their work. This is a common benefit shared across all maturity models and is not explicit to our model, as pointed out by Marnewick and Langerman [34]. However, this participant had no previous experience doing a maturity assessment, and them making this statement led us to believe that even the organizations with high DevOps maturity could benefit from performing a maturity assessment if they haven't done so before.

6.1.2 Points of improvement

One problem the second round of interviews showed us was that the 5 participants noted about the mismatch between the basic level descriptions and the capabilities featured in the model. We have previously argued that both the Eficode model and the capabilities featured in the Accelerate book (and even the practices we have used for our first round of interview with the works of Plant et al. and Offerman et al.) were in-line with the CALMS framework. We believed since all the information was rooted in the same framework there wouldn't be such a mismatch. Furthermore, we tried to make a meaningful mapping between the two categories of capabilities as described earlier. We did not modify the basic level descriptions given in the Eficode's model, since, as also mentioned in section 5.3.3, we did not want to make any direct changes to the Eficode's model itself, because one of the reasons why we chose Eficode as our foundational model was that it was supported by literature, and altering the model would be conflicting with this reason of choice. One potential solution to address this "mismatch problem" could be to add more information to all the capabilities featured in the model by providing basic level descriptions for each of them, (i.e. 4 descriptions for each level, for each capability). This way, the connection between the foundational model and the capabilities will be emphasized. An alternative solution could be to implement the basic level descriptions for each of the capabilities while removing the foundational model altogether, and performing the assessment by giving a rating for each of the capabilities. Another solution is to redesign the Eficode model, or make a foundational model from scratch, in order to make the base model more in-line with the capabilities, and keep the information about the capabilities the same.

The second round of interviews also revealed that 2 participants that worked in an "alternative" DevOps manner, where development and operations teams were not working together as a single team (i.e. the "traditional" or "typical" way of doing DevOps), but there was an intermediate DevOps team that acted as a "bridge" between the two. The insights gathered from these

2 participants not only made the aforementioned "mismatch problem" more pronounced, but also they have made different remarks than the rest of the participants, showing that our model might not be well suited towards individuals working in a DevOps manner like this (i.e. "DevOps as an external party" as mentioned in [22]). The exact manner in which our participants was doing DevOps was of no consideration to our research, so it is hard to reach any meaningful conclusions on which kind of DevOps team structure our model is best suited for.

We claim that the comments on the user experience and user interface can be disregarded, as they were vastly different from each other and were most likely dependent on personal preferences of particular participants.

6.2 Implications of the insights on DevOps practices

6.2.1 What practices are the most popular?

As shown earlier in section 5.3.3, we have created some codings on the DevOps practices that were discussed in the first round of interviews. As it can be seen with the frequencies, the practices related to the "Culture" and "Automation" categories were more extensively discussed compared to "Monitoring" and "Agile/Lean" categories.

With "Culture", the most discussed practice was "self-organizing", with "continuous improvement", "tool selection" and "decision making" sharing a close second. Among these, "self-organizing", "tool selection" and "decision making" are all related to the "self-empowerment" capability as given by Plant et al. [2]. Regarding this capability, some of the participants talked about how they were suffering from "too-much autonomy". As shown earlier, we have tried to address this problem with our model via the suggestions given with related capabilities, and the second round of interviews confirmed that the suggestions were meaningful in order to address it. One important thing to note about this particular issue was that we had not observed this problem explicitly mentioned in the literature we have covered. The literature typically talks about how "traditional" organizations limit autonomy, then how and why the autonomy of teams should be increased; but the problem we identified showed the other extreme [28]. Considering digital transformation and DevOps transformation journeys are a common trend within the software industry nowadays [35], [20]; this problem may appear more and more in the upcoming years within organizations looking to change their cultural practices. As such, future research could seek to explore and address this problem in more detail.

As for "continuous improvement", we have already mentioned about how practitioners believed continuous improvement regarding every dimension was essential in order to progress and mature as an organization. One participant in particular had great comments related to this, stating that "people underestimate 2 things, the amount of change that is necessary and the amount of change that's possible" and that "continuous improvement is not only possible, it's necessary".

With "Autonomy", the most discussed practice was "containerization". Participants who have been using containers or sought to integrate containers into their work within the near future, talked about just how much the speed of their deployment increases with the use of containers. The literature is also in-line with these remarks of participants [3], [26]. The shared belief between literature and practitioners is that containerization is the most useful practice to implement regarding continuous and/or automated deployment, until the "new best thing" comes to existence and replaces containers. Containerization is mentioned in the model on the capability specific page of "automated deployment".

With "Agile/Lean", there weren't any big difference with the most discussed practices, but for the least discussed, "eliminating waste" stood out among the "Agile/Lean" practices. Fur-

thermore, the use of "Kanban boards" shared the second place with "incident handling", as the least discussed practice among "Agile/Lean" practices. Lean is significantly characterized by eliminating waste and Kanban boards, so considering that, it's reasonable to assume that our set of participants did not identify with the Lean as strongly as they could identify with other Agile methodologies (e.g. Agile defined by the "Agile manifesto", Scrum). It's hard to draw definite conclusions from this, as the "least discussed" topics may have been discussed less simply because some other topics were more interesting to talk about for the participants, but there is a probability that the DevOps practitioners of today do not strongly identify with the Lean methodology.

With "Monitoring", there weren't any big differences among the practices that were discussed when it came to frequency.

6.2.2 Adoption and Evolution of practices

In the first round of interviews, we had talked to our participants about the evolution, the adoption and the order of importance of related to the practices. We have previously stated that the shared opinion of our participants was that all the practices are meant to be adopted and evolved together. This actually turned out to be a little surprising, because one popular view of DevOps, both in scientific and gray literature, primarily defines DevOps as a cultural movement, and argues that developing cultural practices is a foundation for other aspects of DevOps (i.e. "dimensions" as defined in CALMS) to be built on [40], [41]. Our participants sharing the belief that continuous improvement regarding every dimension is necessary to establish DevOps practices shows that this "culture as a foundation" view might be inaccurate.

It is perhaps more accurate to view each dimension of CALMS as equally important and try to develop capabilities related to them at the same time. This not only ensures that organizations are not experiencing any bottlenecks in their performances by a single dimension, but according to one of our participants, doing so "creates more opportunities for synergies", where the positive intricate relationships between capabilities are realized.

In the model, we tried to highlight such relationships we could identify, using our interviews and the literature; but to fully show these relationships and what the benefits of realizing them would bring, the capabilities should be analyzed with a stronger focus, utilizing the "multilevel" description of capabilities, as shown earlier within the paper [36] [4].

With regards to our sample, we have argued that the level of seniority was of no concern to our research, but the years of experience within their current work function and the organization was. The reasoning for this is that the participants needed some experience to learn about their organizations' capabilities, and this learning could be done at any seniority level. This approach indeed turned out to be true, as the participants who had more years of experience within their work function and their organization was more confident in talking about the state of the capabilities compared to those who had less years of experience; even though the former had less total years of work experience (i.e. lower level of seniority) than the latter. That's not to say that the "less confident" individuals made no meaningful remarks, it's just that they were more hesitant in talking about subjects that required prior information they didn't possess (e.g. with evolution of practices, you need to have a good understanding of the previous state of the practices to talk about what they evolved into). As stated earlier, all the participants have at least 1 year of experience within their current work function and organization.

6.2.3 Order of importance of practices

We have previously stated that there were differences as to which dimension of CALMS and the practices associated with these dimensions the participants prioritized when it came to adopting/improving them. For example, we have stated that the two participants working in an "alternative" DevOps team structure (where their DevOps team acted as "bridge" between separate "dev" and "ops" teams) emphasized how important monitoring and some of the associated practices (e.g. centralized logging) was for them. Overall, the rest of the participants did not comment on this importance as strongly, but there were differences as to what they prioritized the most. As an example, a participant put their highest importance on practices related to automation dimension (e.g. automated deployment), stating that the reliability of their deployments was the most important aspect of their work.

These differences in priorities can be attributed to various factors. Such factors that was apparent for us on an organizational level were differences in industry, organization size, team size, team structure, technologies and tools used, and the location of the organization; and on an individual level there were differences in seniority, years of experience within the organization, work function, and years of experience within the function. Although we have covered some of these in our sample description, we cannot disclose details regarding all of these factors to keep our participants data anonymous. However, the deeper reasons as to why such differences in priorities exist, whether it relates to practitioners' team structure, and why these two participants strongly prioritized monitoring remains to be seen.

To summarize, the various settings different organizations operate in makes them have different priorities with their DevOps journey; but they can still unite on the fact that continuous improvement regarding all dimensions is needed. This finding is also in line with the findings described in the Accelerate book [3]. We believe this is why all our participants could relate to the information given in the model despite their differences; and it's interesting to think about whether they could still relate to the information just as well, if we used another DevOps maturity model that didn't utilize capabilities.

6.2.4 Approach to tools

In our insights from our first round we have argued that the approach of participants to tools might be relevant to maturity. Further research into literature showed that this was indeed the case, the particular tools that are used by high-performers are talked extensively on the paper by Blinde [31]. Despite that fact, we did not feature any tool by name in our model, because specific tools are tied with certain best practices, but there are always alternatives when it comes to tooling [31]. Also, as seen, different organizations have different needs, so a certain tool that is useful for a particular organization might not be useful for another.

6.3 Answering the research question

The main research question was established as:

"RQ: How can the lens of capabilities (including dynamic capabilities) improve current DevOps maturity models?"

We have introduced 5 guiding questions in order to make the RQ easier to answer.

"GQ1: What are the organizational capabilities and routines that are related to DevOps?"

In order to answer GQ1, we have thoroughly analysed the available literature on DevOps, as shown in the Literature chapter. As mentioned multiple times throughout the paper and emphasized in section 5.3.3, the works of Offerman et al. and the works of Plant et al. was the most notable sources in order to establish our fundamental knowledge of the capabilities and routines related to DevOps [26], [31], [8], [2]. Establishing this fundamental base of knowledge helped us prepare for the first round of interviews as well. Furthermore, a big aspect to this first round of interviews was also in regards to contributing to this fundamental knowledge base; since seeing the perspectives of practitioners brought an understanding of the practical side of DevOps capabilities to our knowledge base.

Understanding which organizational capabilities and routines are used in DevOps wasn't not enough, as it was also necessary to learn what capabilities and routines are utilized by high-performers to be able to discern between the "good" and "bad" practices, technologies, ways of working, and all the other aspects of capabilities and routines; so that meaningful suggestions can be provided within our model. In order to do this, we have looked at the literature; and also created and analysed some codings on our first round of interviews, as emphasized in section 5.3. The most notable literary works that helped us in this regard was the Accelerate book and the DevOps Handbook [3], [28]. The capabilities and practices featured in the model, which originate from these books and the codings can be provided as the answer to the GQ1 (see table 5.2).

"GQ2: How do organizations move from one maturity level to another?"

Answering this question was important to make our model function properly. Our first round of interviews showed that implementing suggestions on how to adopt/improve capabilities could help organizations increase their maturity. However prior to seeing how such suggestions could be featured in a model, we have realized that we needed to select a base model to improve upon. Previous analysis of literature gave us 2 prime candidates of DevOps maturity models to select such a base model: Eficode's model and the model of Feijter et al. [30], [33]. As mentioned in chapter 4, and section 5.3; after analyzing both models based on the information provided by literature and based on the insights we have gathered after the first round, we have decided to improve upon the Eficode's model.

Based on literature and our findings/codings from the first round of interviews, we have designed our model such that the information given about the capabilities are corresponding to the the different levels of dimensions; and the suggestions given related to these capabilities are helpful to increase the DevOps maturity of the organization of the users. The details on the design choices are discussed in section 5.3.3. Our second round of interviews validated that implementing these suggestions could help organizations increase their maturity (i.e. move from one maturity level to another). This validation was further emphasized with GQ3.

"GQ3: How does a maturity model that utilizes organizational capabilities and routines fare in practice?"

We had introduced two primary problems with our research. The first problem was that, in mainstream DevOps maturity models, understanding what capabilities and routines organizations needed to adopt or improve upon wasn't clear. We have referred to this problem as the "actionability problem" throughout the paper and tried to address it with our model. Our second round of interviews showed us that 6 out of 11 participants found the suggestions given in the model helpful. While the remaining 5 participants did not found such suggestions imprac-

tical, they have found them trivial. This means that our model was moderately successful in addressing the actionability problem.

The second problem we introduced was that, according to our analysis of the literature, the mainstream DevOps maturity models weren't made with dynamic capabilities in mind. We tried to explore how dynamic capabilities could be featured in a DevOps maturity model and tried to implement what we have learned into our model. We have tried to make our model "dynamic" as per the information that was found in the literature, and the remarks of participants in the second round confirmed that our model was practically dynamic. We have discussed about these results and their implications in further detail in the Discussion chapter.

We claim that the improved model fared successful in addressing the problems we had introduced, based on the data gathered from the validation round. That being said, our research also had some limitations; and also lead to some unanswered questions and improvement points that could be researched further. All of these are discussed in the next sections.

6.4 Limitations

In addition to the limitations revealed with our model with the second round of interviews as discussed earlier, there are also other limitations with our research.

We have previously featured literature that claimed DevOps maturity models are not sufficient to drive organizational changes [34]. We have indeed confirmed this with the insights we have gathered, and as described in section 5.3.2 earlier, it was argued that all DevOps maturity models should be used with respect to two important factors: regular and periodic use (periods expressed in months), and use with the intention to self-improve. However, we had only a single round of interviews to validate our model. As such, we weren't able to do periodic assessments, and we also didn't get to see whether our participants try to implement the suggestions that they were given; so we are limited in our ability in seeing how our model performs in an ideal practical setting.

One of our key sources for this research, the Accelerate book, also has criticisms of maturity models [3]. In the book, it is claimed that the mainstream maturity models (which do not feature capabilities to an in-depth level) only help organizations reach a 'mature state and "declare themselves done with their journey", while the technology and business landscape continue to evolve [3]. As shown in our insights, we had participants also making remarks parallel to these criticisms, arguing that it's difficult to be completely mature (i.e. level 4 for our model) regarding any dimension since what's considered mature is constantly changing, and that there is always room for improvement with regards to any dimension or capability. This criticism coming from both academic and practical sides shows us that maturity models are inherently not built to last. However it is also known (and also proven with our research) that when used correctly, maturity models do in-fact help organizations improve, so an intermediate solution could be to update maturity models regularly.

Potential threats to validity of research design are another aspect of limitations with our research, since we have also used research design as our methodology. Such threats are expressed as external or internal factors [42]. The external factors that are relevant for us are biases that are potentially involved. Primarily, these biases are sampling bias and non-response bias. As shown earlier, our sample size for the interviews was 10 and 11 people for the first and second rounds respectively, and all the participants except one were working as DevOps Engineers, and one participant was working as a DevOps coach. Furthermore, if we recall that we had response rates of 5.625% and 6.25% for DevOps engineers for the first and second round respectively; and a 2.5% response rate for DevOps coaches, our sample might not be representative of the actual

population, and the non-respondent population might have had different insights we couldn't have captured.

For internal factors, there are two main concerns. The first concern is related to the instrumentation [42]. Ideally the instrumentation should be consistent, and even though we had used the same model for every interview in the second round, there were differences in the capabilities that were covered as part of our discussions with the participants. This was the intended use, as explained in the section 5.3.2, because covering the capabilities that the participants were looking to adopt/improve on, so that they can benefit from the suggestions is the idea behind the model. However, the set of capabilities that fit that description was naturally different for each participant. It remains to be seen whether the results would be the same if we were to cover the same exact set of capabilities in every interview, but doing so would be against the intended use of the model and wouldn't be practical for the participants, so we didn't do it that way. The second internal concern is related to the effects of "testing" on the outcomes of the validation round [42]. As explained, we had discussed about various capabilities and practices in the first round. There is a possibility that some participants may have picked up on the best practices and related capabilities; and during the second round, used that knowledge to get a higher score with the model regardless of whether their organization actually possessed these capabilities and best practices, consciously or subconsciously.

Chapter 7

Conclusion

As part of this research we tried to see how capabilities could be integrated into a DevOps maturity model and established our research question as *"How can the lens of capabilities improve current DevOps maturity models?"*. Capabilities were integrated into a DevOps maturity model such that the information about them are provided in a manner where first, they are described; second, reasons as to why these capabilities should be present in organizations are given; and third, practical suggestions such as how to establish these capabilities in the short term, how they should be like in the long term, best practices and common pitfalls are given.

The capabilities in the model fall under 4 "capability categories": ("Cultural", "Continuous Delivery", "Architectural", "Lean Management and Monitoring"). These categories correspond to 6 dimensions of the original model (Eficode's model). These dimensions are "Organization and Culture", "Environments and Release", "Builds and Continuous Integration", "Quality Assurance", "Visibility and Reporting", and "Technologies and Architecture".

A design research methodology was used to develop and validate this model. As part of this, a first round of interviews was conducted to learn about the DevOps practices and approach to maturity assessments of DevOps practitioners. Following these interviews, Eficode's DevOps maturity model was selected as the base model to improve upon and using the data gathered from the interviews and literature, the improvements were implemented on this model. A second round of interviews was carried out to validate the model, and it was found that the model was moderately successful in showing practitioners which capabilities and routines their organizations needed to adopt or improve in order to increase their maturity, and how such improvements/adoptions can be made (this problem was mentioned as the "actionability problem"). It was also found that the model was easy to use and was practically "dynamic".

There were also problems revealed with the second round of interviews. Firstly, connection between the original model and the capabilities featured in it didn't match perfectly; secondly, it was found that the information on some of the capabilities weren't complete; lastly, the model was found to be not suited perfectly for DevOps practitioners who were doing DevOps using an "alternative" team structure, where the DevOps team acted a bridge between development and operations teams.

There were also other findings regarding maturity assessments and DevOps practices.

For maturity assessments, it is found that the maturity assessments lead to improvements only when two criteria regarding the assessment is satisfied. The first criteria is regarding doing multiple assessments, periodically. The periods should be expressed in months, not years, and the assessments should be scheduled and executed with no/minimal delays. The second criteria is having an intention to improve, where certain goals are set as a result of the assessment; and

when the assessment is performed, seeing whether these goals are met and whether the maturity scores on the dimensions related to these goals have improved from the last time the assessment was performed. When the assessment is done a single time, or with long/spontaneous periods; or when there is no intention to "beat your personal best", doing maturity assessments (with or without the use of a maturity model) does not lead to any practical benefits such as improvements to organizational performance.

For DevOps practices, it is found that the capabilities and practices related to the "Culture" dimension of DevOps do not act as a foundation for the other dimensions (dimensions as per the definition of CALMS framework are "Culture", "Automation", "Lean", "Measuring/Monitoring", "Sharing"), which is contrary to what some experts argue [40]. Instead, our findings show that all the dimensions are related to one another and the practices related to them are meant to be adopted and evolved together.

"Self-organizing", "continuous improvement" and "containerization" were found to be some of the most popular practices related to DevOps. For "self-organizing" in particular, multiple participants from our interviews talked about their experience having "too much autonomy", which was interesting since the literature usually talks about how and why organizations should seek to increase the autonomy of their teams [3]. Practices associated with the Lean methodology were the least discussed within our interviews, which may suggest that the DevOps practitioners are straying away from the Lean methodology.

When it comes to the order of importance of DevOps practices, it was found that the aforementioned DevOps practitioners who were doing DevOps using an "alternative" team structure strongly prioritized monitoring the most. While the rest of the participants' opinions differed, they did not comment on this importance as strongly. These differences in priorities likely depend on both organizational factors (e.g. industry) and individual factors (e.g. personal experiences). The more in-depth reasons as to why such differences in priorities exist were not a part of this research.

To conclude, capabilities were found to be a practical solution to improve a DevOps maturity model, enabling practitioners to access specific information regarding different aspects of software development easily. The research and the model that was created was mostly successful in addressing the problems that were introduced, but it also generated some questions and some improvement points related to the model were revealed, which could be researched further.

7.1 Future Work

As mentioned multiple times throughout this paper, our insights revealed that maturity models should be used periodically and with "an intent to improve", but as described earlier, we didn't do that as part of our research. Furthermore, in our "Methodology" chapter, we have argued that to get a deep, multilevel understanding of routines and capabilities in a particular organization, data should be collected over a long period of time; but that we did not intend to do that as we only needed to focus on the ostensive aspect of capabilities for our research [36], [6]. In the light of these findings and the literature, it would be certainly interesting to see how our model holds up in practice, when it is used in an empirical study within a single organization over a long period, multiple times, and with an intent to improve. Seeing the performative aspect of capabilities for that particular organization and how that performative aspect relates to our model and the capabilities featured within it would be a great progression for this research. Another aspect to such a research could be to identify and explain the complex relationship between capabilities. Even though we tried to highlight all the connections we could identify between the capabilities, doing such a study and establishing a deeper understanding of capabilities could make these

connections even more pronounced and might even bring up new connections that we couldn't identify with our research.

Overall, we believe that our model is a better and improved version of the Eficode's model and could serve as an example to those who develop maturity models, especially on how to feature capabilities.

Even though our model can be deemed as successful, it's not flawless. As such, another research topic could be to improve our model even further. We have already mentioned about some potential improvement ideas in the previous chapter. One of those ideas is regarding expanding on capabilities by refining the information provided within the model so that more details about the capabilities are covered, and the model is more "complete" as a result. Another improvement idea is to try and address the aforementioned "mismatch problem"; we have already mentioned about some potential solutions that could be implemented to address it earlier. Yet another improvement idea is exploring how the model could be altered for DevOps teams working with "alternative" DevOps team structures (see [22]), since we discussed that our model might not be ideal for such teams.

As described in the limitations, both the literature and our findings indicate that the "ever-changing nature of technology and business landscape", makes DevOps maturity models outdated over time [3]. Researching on the best way to make maturity models remain up-to-date with the latest practices and technologies; and trying to find how capabilities should evolve within maturity models could be another interesting topic.

Bibliography

[1] Christof Ebert et al. "DevOps". In: *Ieee Software* 33.3 (2016), pp. 94–100.

[2] Olivia H Plant, Jos van Hillegersberg, and Adina Aldea. "Design and Validation of a Capability Measurement Instrument for DevOps Teams". In: *International Conference on Agile Software Development*. Springer, Cham. 2022, pp. 151–167.

[3] Jez Humble and Gene Kim. *Accelerate: The science of lean software and devops: Building and scaling high performing technology organizations*. IT Revolution, 2018.

[4] Carlo Salvato. "Capabilities unveiled: The role of ordinary activities in the evolution of product development processes". In: *Organization science* 20.2 (2009), pp. 384–409.

[5] David J Teece, Gary Pisano, and Amy Shuen. "Dynamic capabilities and strategic management". In: *Strategic management journal* 18.7 (1997), pp. 509–533.

[6] Martha S Feldman and Brian T Pentland. "Reconceptualizing organizational routines as a source of flexibility and change". In: *Administrative science quarterly* 48.1 (2003), pp. 94–118.

[7] Oliver Schilke, Songcui Hu, and Constance E Helfat. "Quo vadis, dynamic capabilities? A content-analytic review of the current state of knowledge and recommendations for future research". In: *Academy of management annals* 12.1 (2018), pp. 390–439.

[8] Olivia H Plant, Jos van Hillegersberg, and Adina Aldea. "How DevOps capabilities leverage firm competitive advantage: A systematic review of empirical evidence". In: *2021 IEEE 23rd Conference on Business Informatics (CBI)*. Vol. 1. IEEE. 2021, pp. 141–150.

[9] Mohammad Zarour et al. "A research on DevOps maturity models". In: *Int. J. Recent Technol. Eng* 8.3 (2019), pp. 4854–4862.

[10] Karim Sabbagh et al. "Maximizing the impact of digitization". In: *The global information technology report* 2012 (2012), pp. 121–133.

[11] Michael Gebhart, Pascal Giessler, and Sebastian Abeck. "Challenges of the digital transformation in software engineering". In: *ICSEA 2016* 149 (2016).

[12] Mohammad Khoshgoftar and Omar Osman. "Comparison of maturity models". In: *2009 2nd IEEE International Conference on Computer Science and Information Technology*. IEEE. 2009, pp. 297–301.

[13] Maximilian Röglinger, Jens Pöppelbuß, and Jörg Becker. "Maturity models in business process management". In: *Business process management journal* 18.2 (2012), pp. 328–346.

[14] G Bou Ghantous and Asif Gill. "DevOps: Concepts, practices, tools, benefits and challenges". In: *PACIS2017* (2017).

[15] Stuart Galup, Ronald Dattero, and Jing Quan. "What do agile, lean, and ITIL mean to DevOps?" In: *Communications of the ACM* 63.10 (2020), pp. 48–53.

[16] *What is agile?* URL: https://learn.microsoft.com/en-us/devops/plan/what-is-agile.

[17] Ahmed Bahaa Farid, Yehia Mostafa Helmy, and Mahmoud Mohamed Bahloul. "Enhancing lean software development by using DevOps practices". In: *International Journal of Advanced Computer Science and Applications* 8.7 (2017).

[18] Jez Humble, Joanne Molesky, and Barry O'Reilly. *Lean enterprise.* " O'Reilly Media, Inc.", 2020.

[19] Jason Sharp and Jeffry Babb. "Is Information Systems late to the party? The current state of DevOps research in the Association for Information Systems eLibrary". In: (2018).

[20] Ina M Sebastian et al. "How big old companies navigate digital transformation". In: *Strategic information management*. Routledge, 2020, pp. 133–150.

[21] Jez Humble and Joanne Molesky. "Why enterprises must adopt devops to enable continuous delivery". In: *Cutter IT Journal* 24.8 (2011), p. 6.

[22] Shana Vu. *The importance of devops team structure*. URL: https://www.atlassian.com/devops/frameworks/team-structure.

[23] Jez Humble and David Farley. *Continuous delivery: reliable software releases through build, test, and deployment automation*. Pearson Education, 2010.

[24] Erik van Ommeren et al. "Design to Disrupt". In: ().

[25] Ian Buchanan. *CALMS framework*. URL: https://www.atlassian.com/devops/frameworks/calms-framework.

[26] Tyron Offerman et al. "A Study of Adoption and Effects of DevOps Practices". In: *arXiv preprint arXiv:2211.09390* (2022).

[27] Jens Smeds, Kristian Nybom, and Ivan Porres. "DevOps: a definition and perceived adoption impediments". In: *Agile Processes in Software Engineering and Extreme Programming: 16th International Conference, XP 2015, Helsinki, Finland, May 25-29, 2015, Proceedings 16*. Springer. 2015, pp. 166–177.

[28] Gene Kim et al. *The DevOps handbook: How to create world-class agility, reliability, & security in technology organizations*. IT Revolution, 2021.

[29] Tobias Mettler. "Maturity assessment models: a design science research approach". In: *International Journal of Society Systems Science* 3.1-2 (2011), pp. 81–98.

[30] Eficode. *Devops: Eficode quick guide*. 2020. URL: https://www.eficode.com/hubfs/documents/Eficode-English-Devops-Guide.pdf?hsLang=en.

[31] Robert Blinde. "DevOps Unravelled: A Study on the Effects of Practices and Technologies on Organisational Performance". In: (2022).

[32] Ineta Bucena and Marite Kirikova. "Simplifying the DevOps Adoption Process." In: *BIR Workshops*. 2017, pp. 1–15.

[33] Rico Feijter et al. "Towards the adoption of DevOps in software product organizations: A Maturity Model Approach". In: *Technical Report Series* UU-CS-2017-009 (2017).

[34] Carl Marnewick and Josef Langerman. "DevOps and organizational performance: The fallacy of chasing maturity". In: *IEEE Software* 38.5 (2020), pp. 48–55.

[35] Shahid Ali et al. "DevOps in Practice a Systematic Literature Review". In: *Journal of Information & Communication Technology (JICT)* 16.1 (2022).

[36] Carlo Salvato and Claus Rerup. "Beyond collective entities: Multilevel research on organizational routines and capabilities". In: *Journal of management* 37.2 (2011), pp. 468–490.

[37] David J Teece. "Business models and dynamic capabilities". In: *Long range planning* 51.1 (2018), pp. 40–49.

[38] Bram Klievink and Marijn Janssen. "Realizing joined-up government — Dynamic capabilities and stage models for transformation". In: *Government information quarterly* 26.2 (2009), pp. 275–284.

[39] Nigan Bayazit. "Investigating design: A review of forty years of design research". In: *Design issues* 20.1 (2004), pp. 16–29.

[40] Mandi Walls. *Building a DevOps culture.* " O'Reilly Media, Inc.", 2013.

[41] *Introduce the foundation pillars of DevOps.* URL: https://learn.microsoft.com/en-us/training/modules/introduce-foundation-pillars-devops/.

[42] Chong-ho Yu and Barbara Ohlund. *Threats to validity of research design.* 2010.

Appendix A

Questions used in the interviews

A.1 Questions for the first cycle

A.1.1 General questions

1. Can you introduce yourself?
2. Ask the following questions if not already answered:
   - What department are you in?
   - What is your function?
   - How long have you been working for this organization?
   - How long have you been working in your current function?

A.1.2 Practices

The following questions will be asked utilizing practices featured in the works of Plant et al. and Offerman et al. (see figure 2.3 and table 2.1)

1. Which of these practices do you also follow within your organization?
2. How would you assess your maturity level with these practices?
3. What practices did you first adopt and develop? What did these practices evolve into?
4. In what order did you develop these practices?
5. How did you embed these practices then?
6. How do you embed new practices now? Is there a dedicated process for adoption of new practices?

A.1.3 Experience with maturity models

1. Have you used a maturity model (for DevOps or for another use) before? If so, please comment on your experience.
2. What were/are the requirements for selecting and using a model?

3. Ask the following if not already answered:
    - Who executes the assessment of maturity?
    - Is the assessment on a team-level or for the entire organization?

A.2 Questions for the second cycle

The questions for the second cycle were created following the creation of the maturity model. The model is described in the "Results" chapter.

A.2.1 General questions

These questions will be asked if the participant is not familiar with the research (i.e. they were not part of the first round of the interview). If they are, the interview will start with the next set of questions.

1. Can you introduce yourself?
2. Ask the following questions if not already answered:
    - What department are you in?
    - What is your function?
    - How long have you been working for this organization?
    - How long have you been working in your current function?

A.2.2 Maturity Model - Assessment

Using the maturity model we have created, participant will be guided and their maturity will be assessed for each dimension. In order to do this, first we will discuss about the various capabilities related to a particular dimension, then we will ask the participants to assess themselves regarding that dimension; we then move on to another dimension and repeat the process until all of dimensions are covered. For more information, see the "How to use the model" section (section 5.3.2). Following are the questions that will be asked for each dimension: (As a reminder, dimensions as featured in Eficode maturity model are: "Organization and Culture", "Environments and Release", "Builds and Continuous Integration", "Quality Assurance", "Visibility and Reporting", "Technologies and Architecture").

1. What are some of the practices you do related to this dimension? (If participant also participated in the first round, then we already have existing information about their practices and may skip this question for some dimensions). After learning about their practices (if we don't already know about them from the first round), we intend to talk about capabilities that they may be lacking or need improving.
2. Which capabilities regarding this dimension are you looking to adopt/ improve? How are you going to do this? (As part of the maturity model we will already have some suggestions, so the intent behind this question is about setting a basis for discussion. We also guide the participants who have also participated in the first round towards some specific capabilities, as we have prior knowledge on what they can benefit from the most).
3. Which of these level descriptions do you relate to the strongest, regarding this dimension?

A.2.3 Maturity Model - Feedback

1. Do you think implementing the suggestions provided for specific capabilities will bring value to your organization (and improve your maturity)?
2. Do you think the suggestions given in the model are reasonable to implement for your case? If not, why?
3. What can you recommend to improve on this model?

Appendix B

Improved Maturity Model

Figure B.1: First page of the model (Eficode)

Figure B.2: List of Cultural capabilities

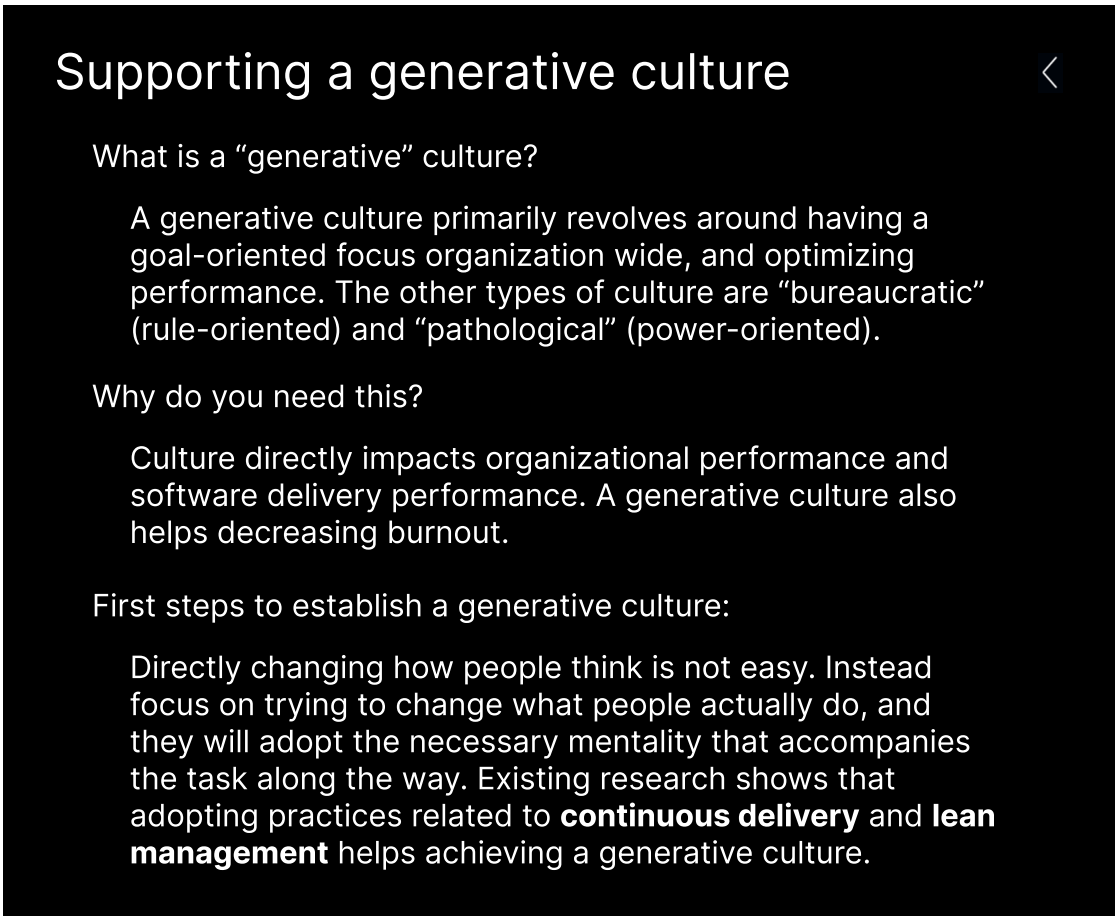

Figure B.3: Cultural capability 1 - Supporting a generative culture

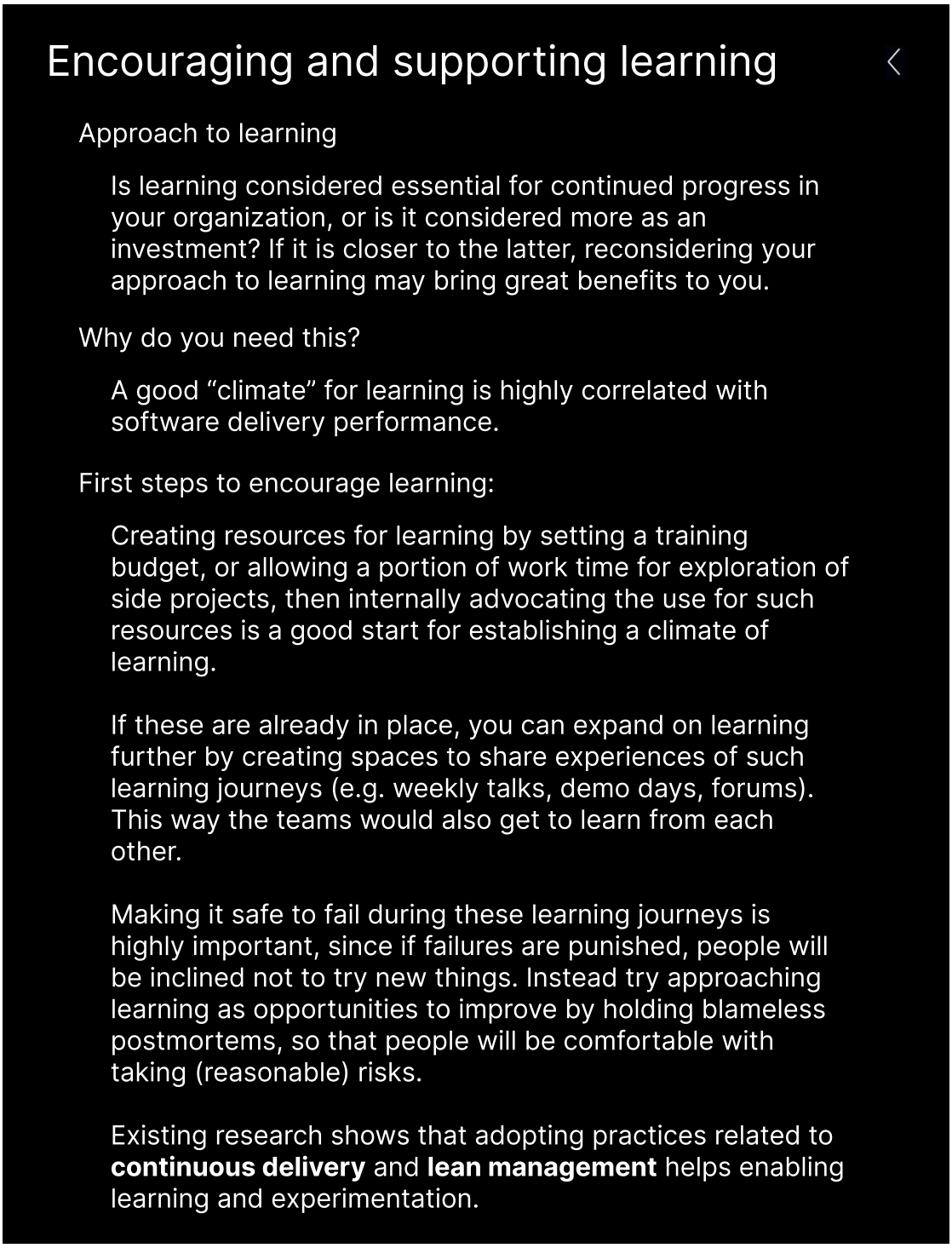

Figure B.4: Cultural capability 2 - Encouraging and supporting learning

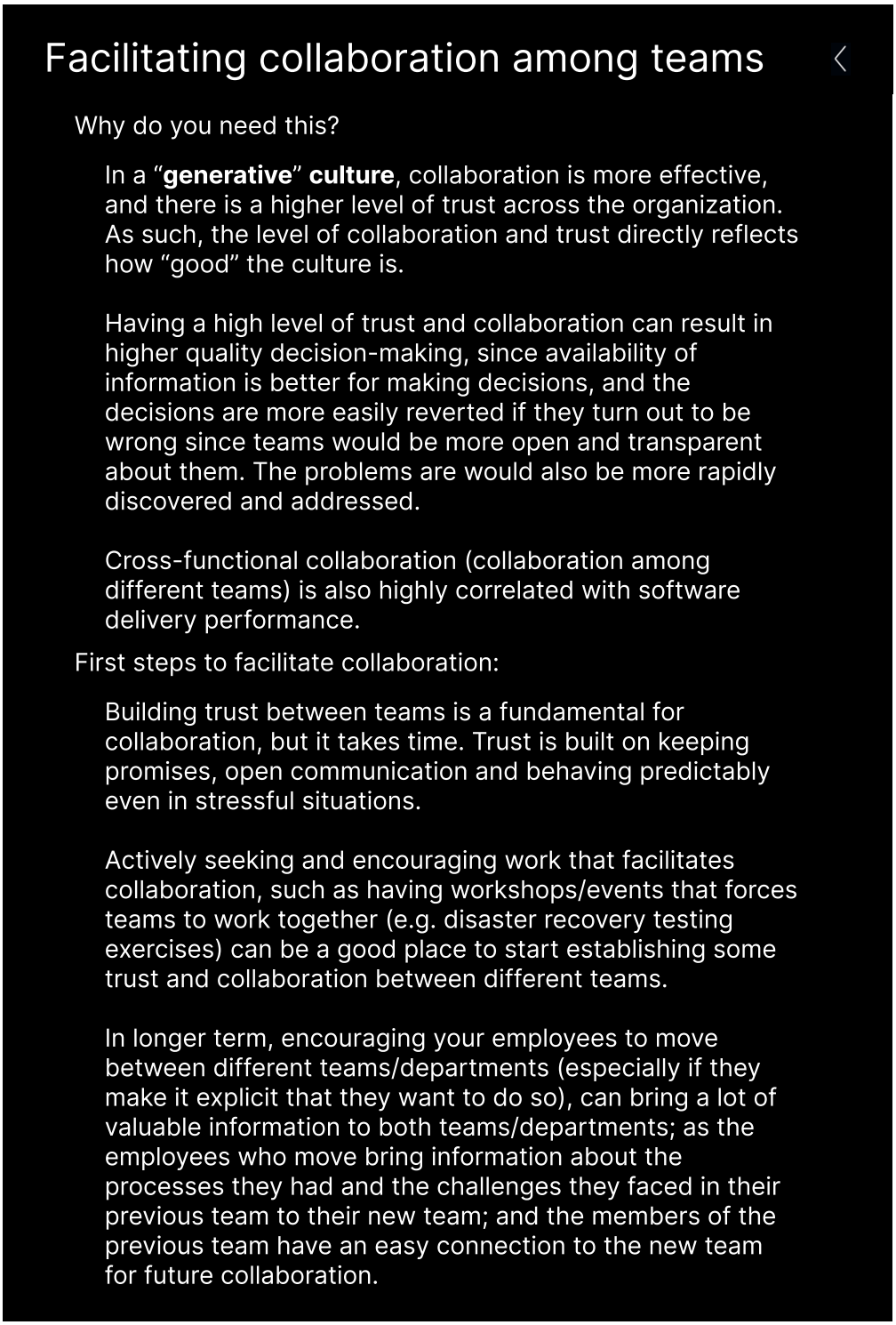

Figure B.5: Cultural capability 3 - Facilitating collaboration among teams

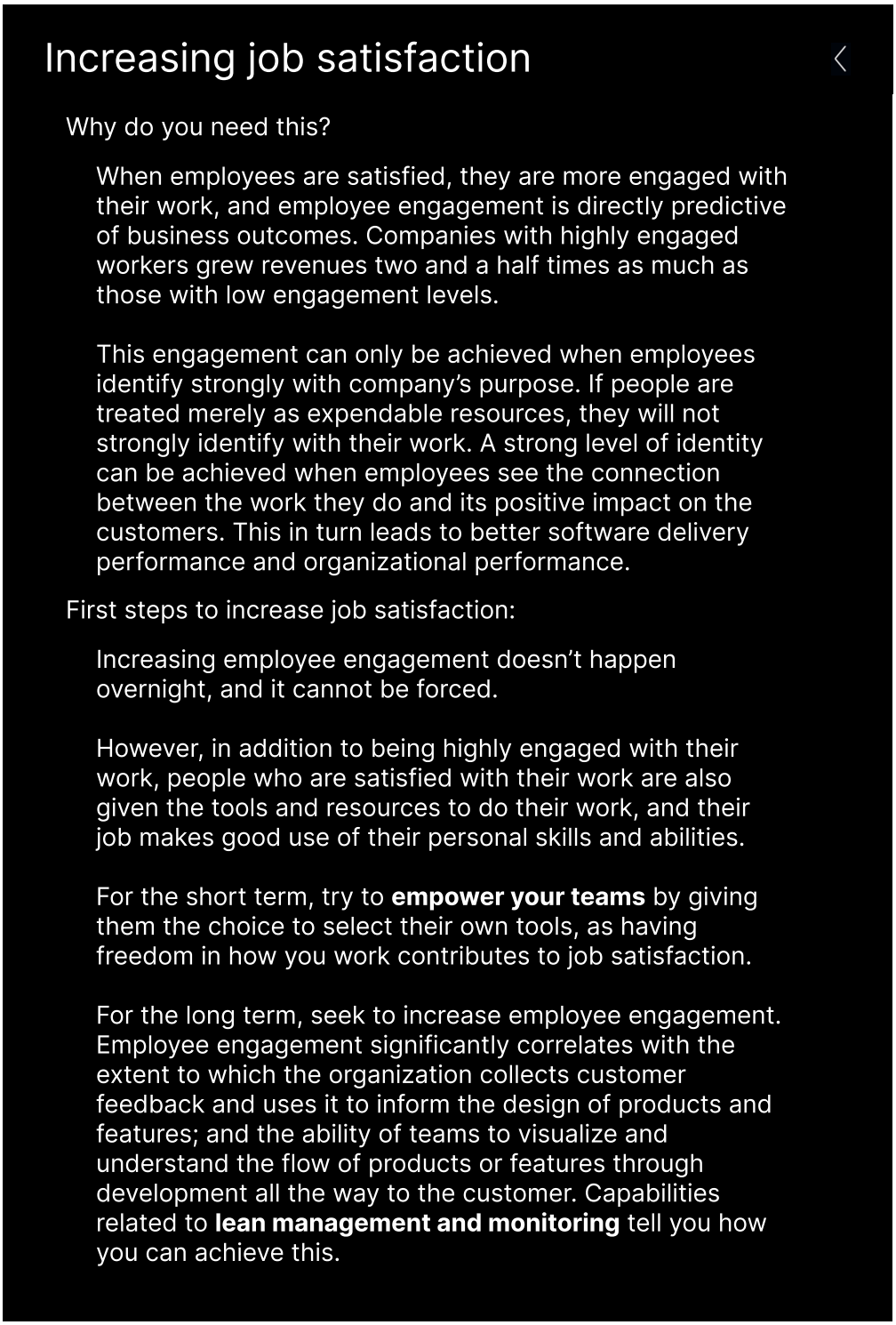

Figure B.6: Cultural capability 4 - Increasing job satisfaction

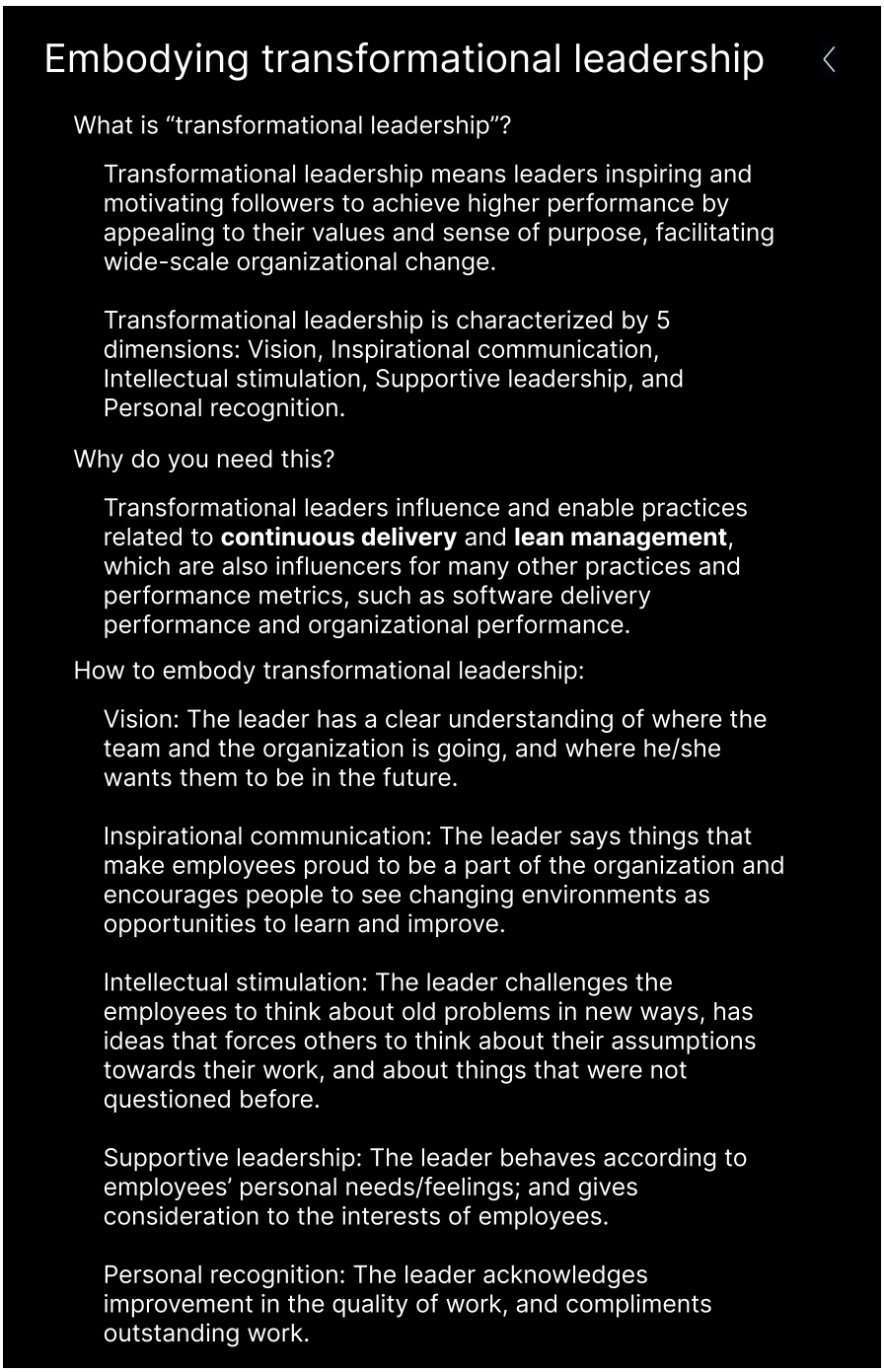

Figure B.7: Cultural capability 5 - Embodying transformational leadership

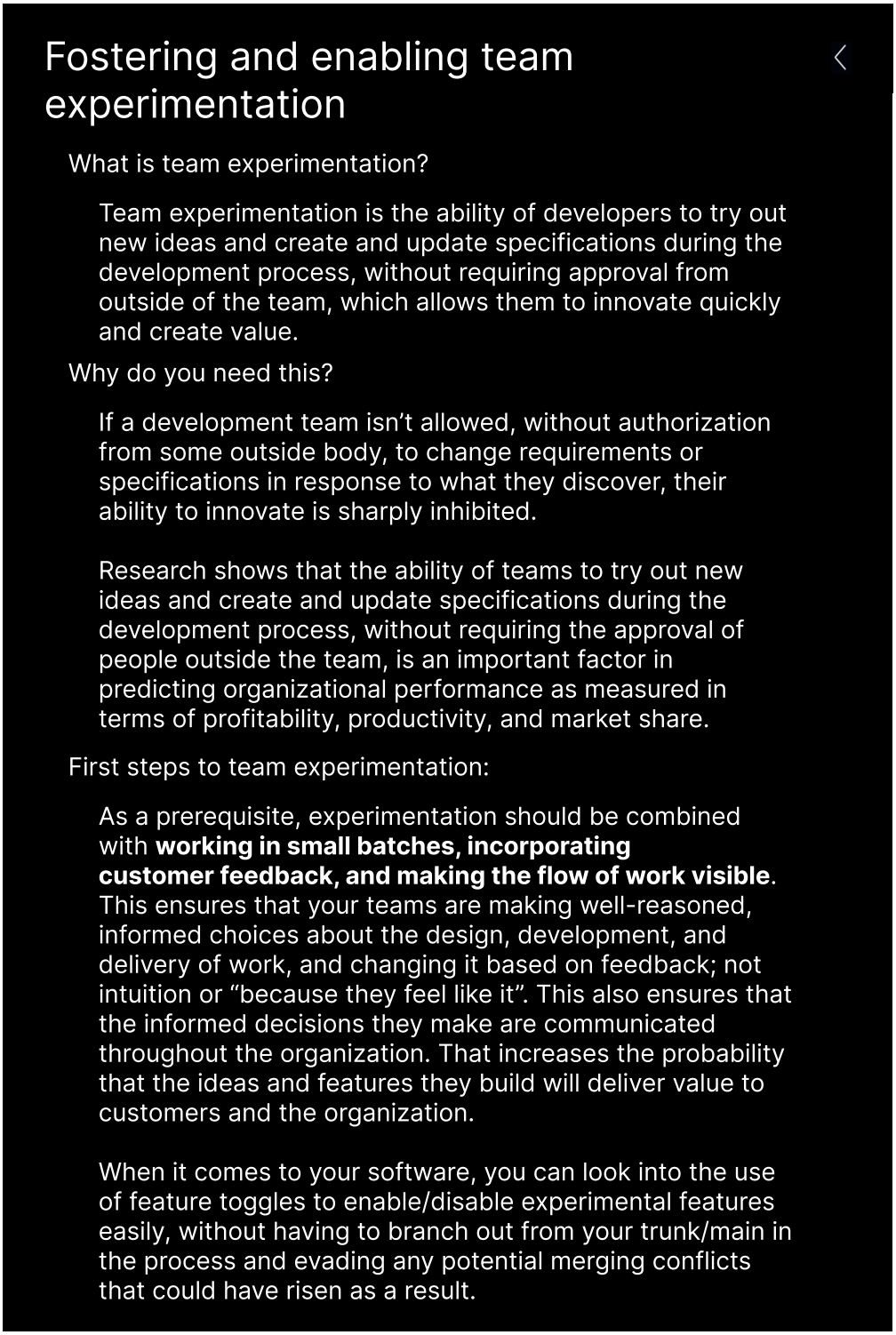

Figure B.8: Cultural capability 6 - Fostering and enabling team experimentation

Figure B.9: List of Continuous Delivery capabilities

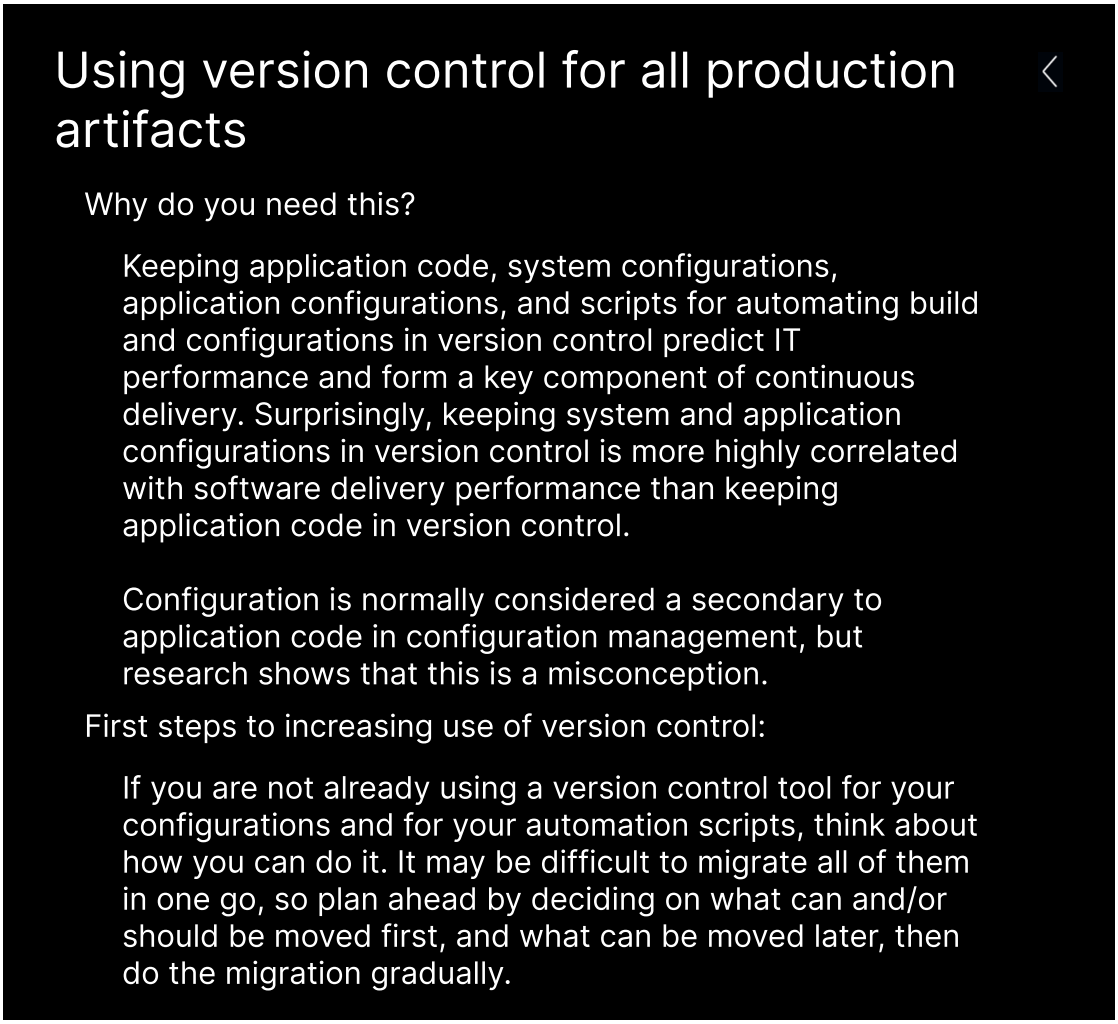

Figure B.10: Continuous Delivery capability 1 - Using version control for all production artifacts

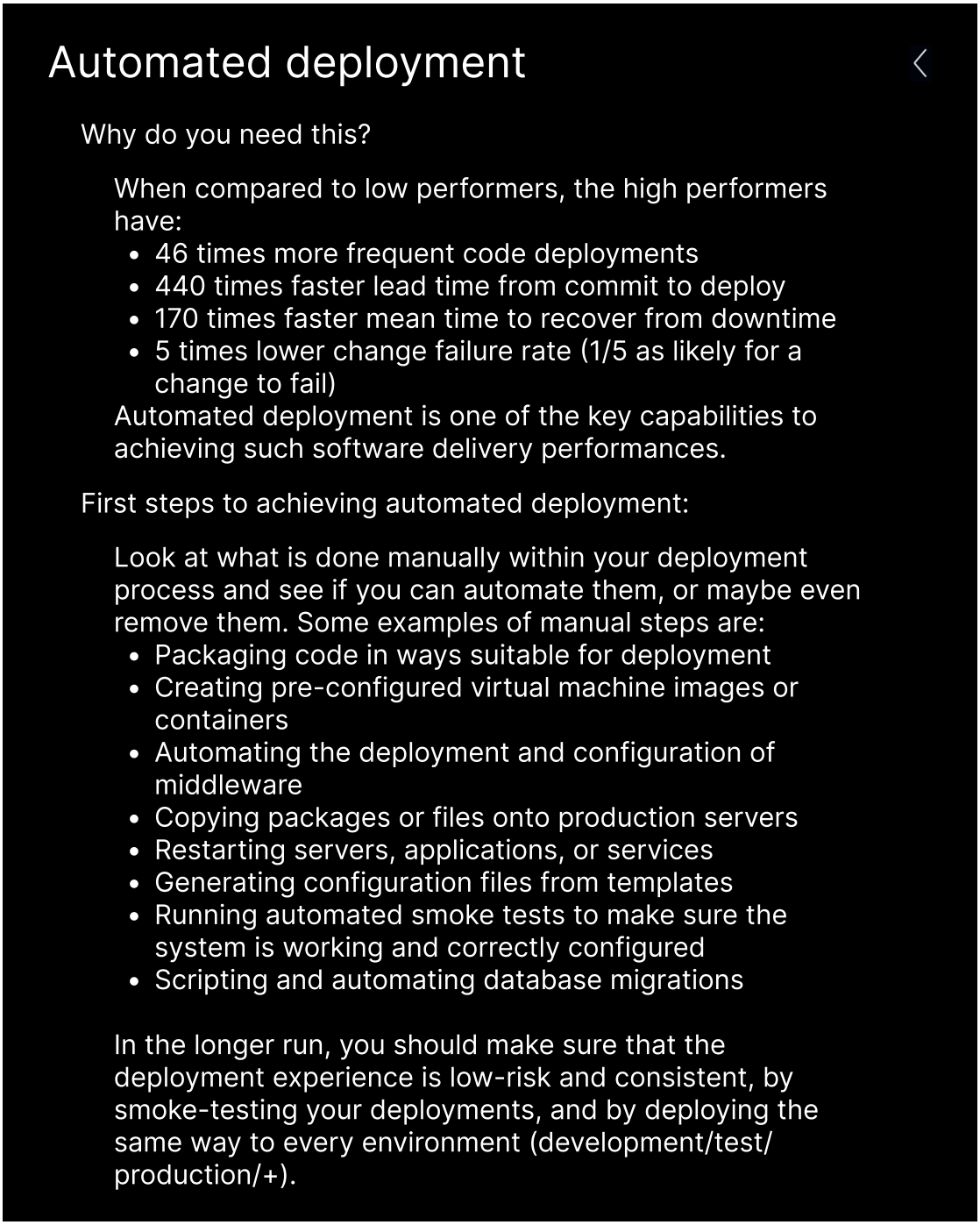

Figure B.11: Continuous Delivery capability 2 - Automated deployment

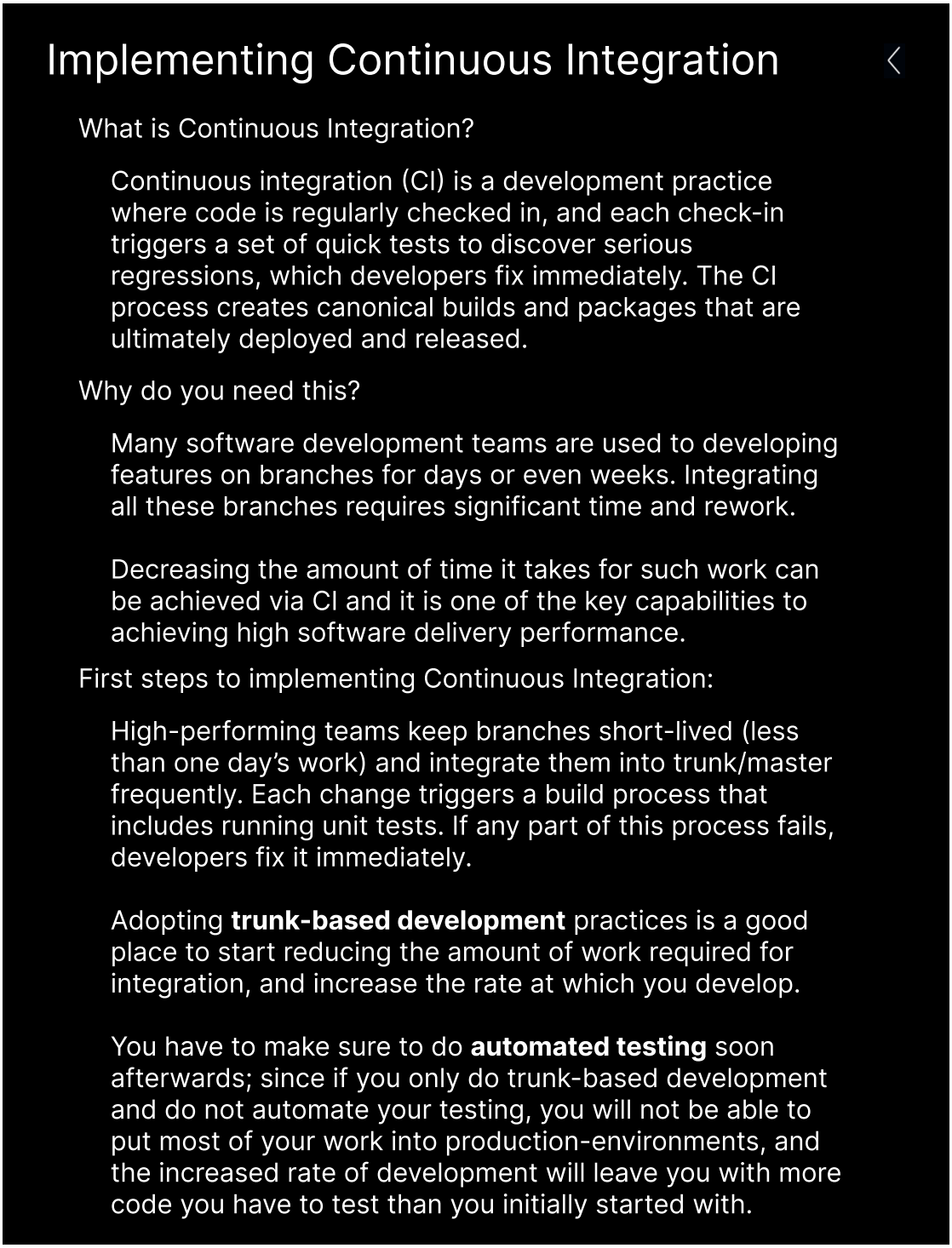

Figure B.12: Continuous Delivery capability 3 - Implementing Continuous Integration

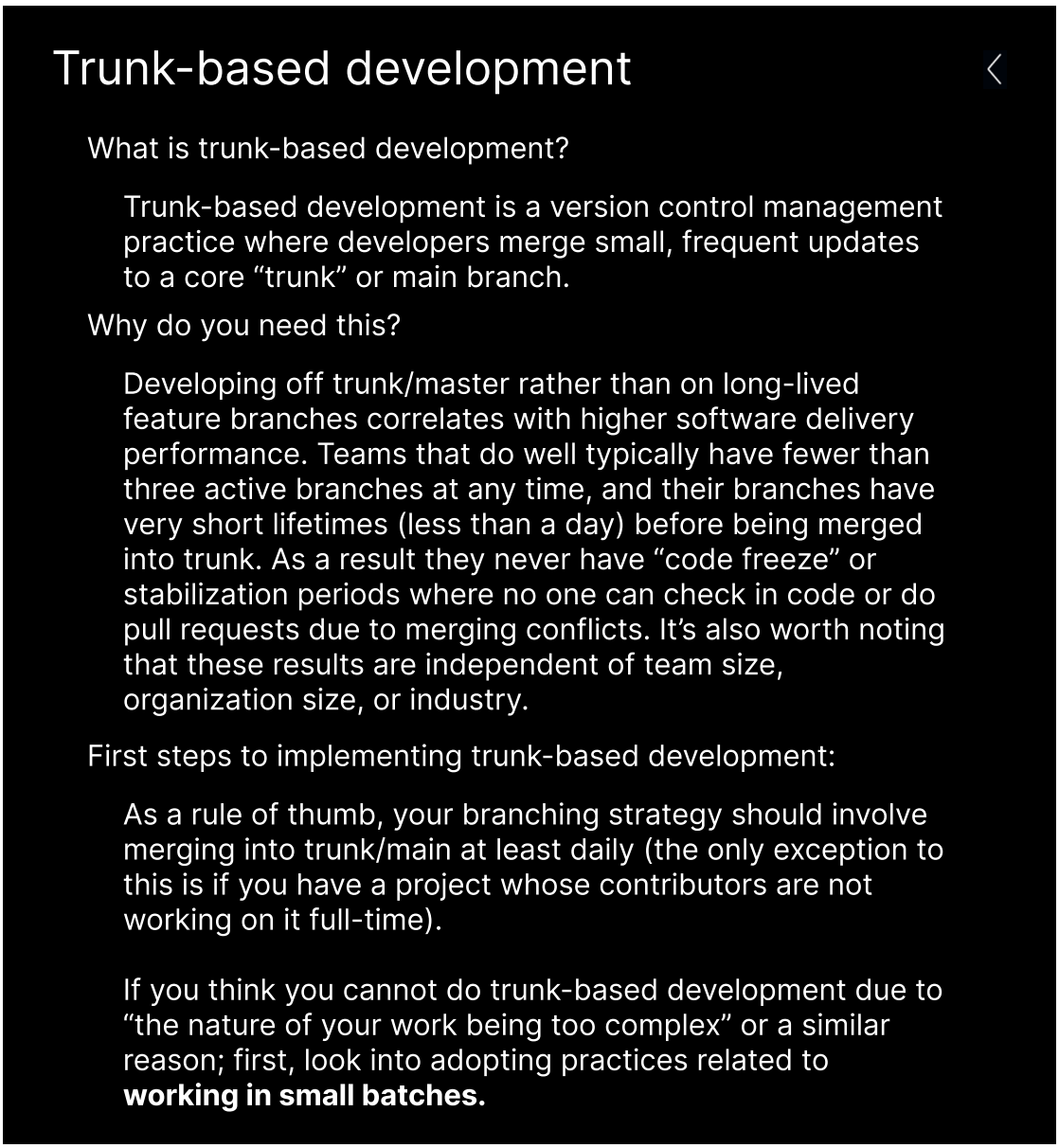

Figure B.13: Continuous Delivery capability 4 - Trunk-based development

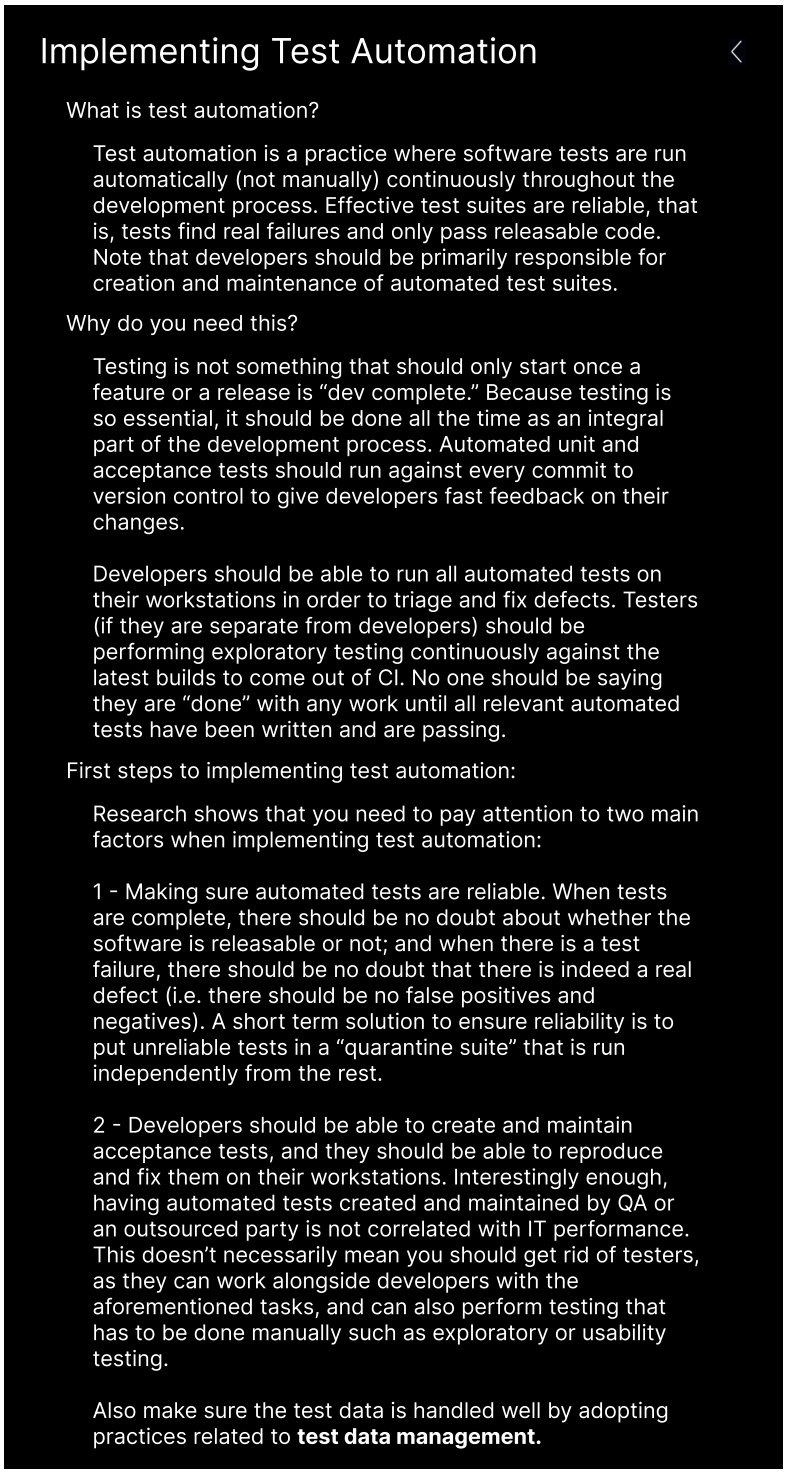

Figure B.14: Continuous Delivery capability 5 - Implementing Test Automation

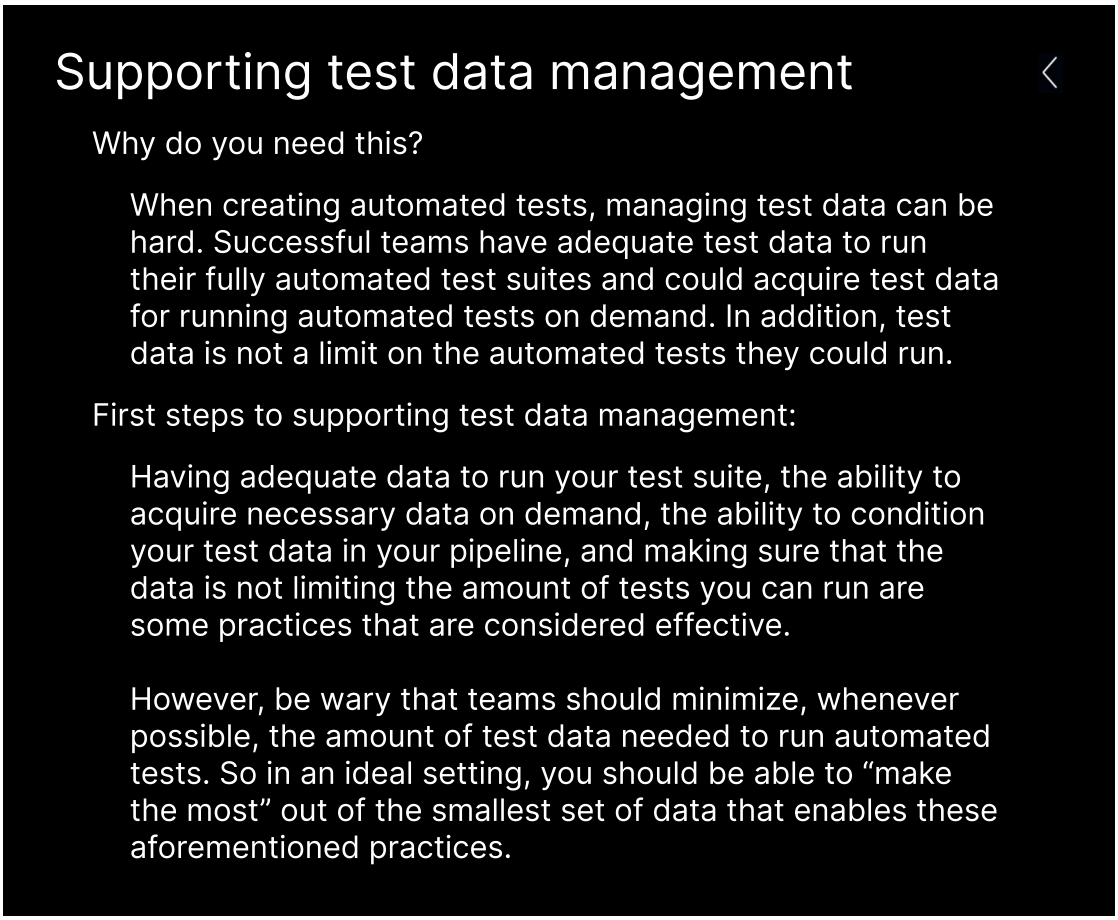

Figure B.15: Continuous Delivery capability 6 - Supporting test data management

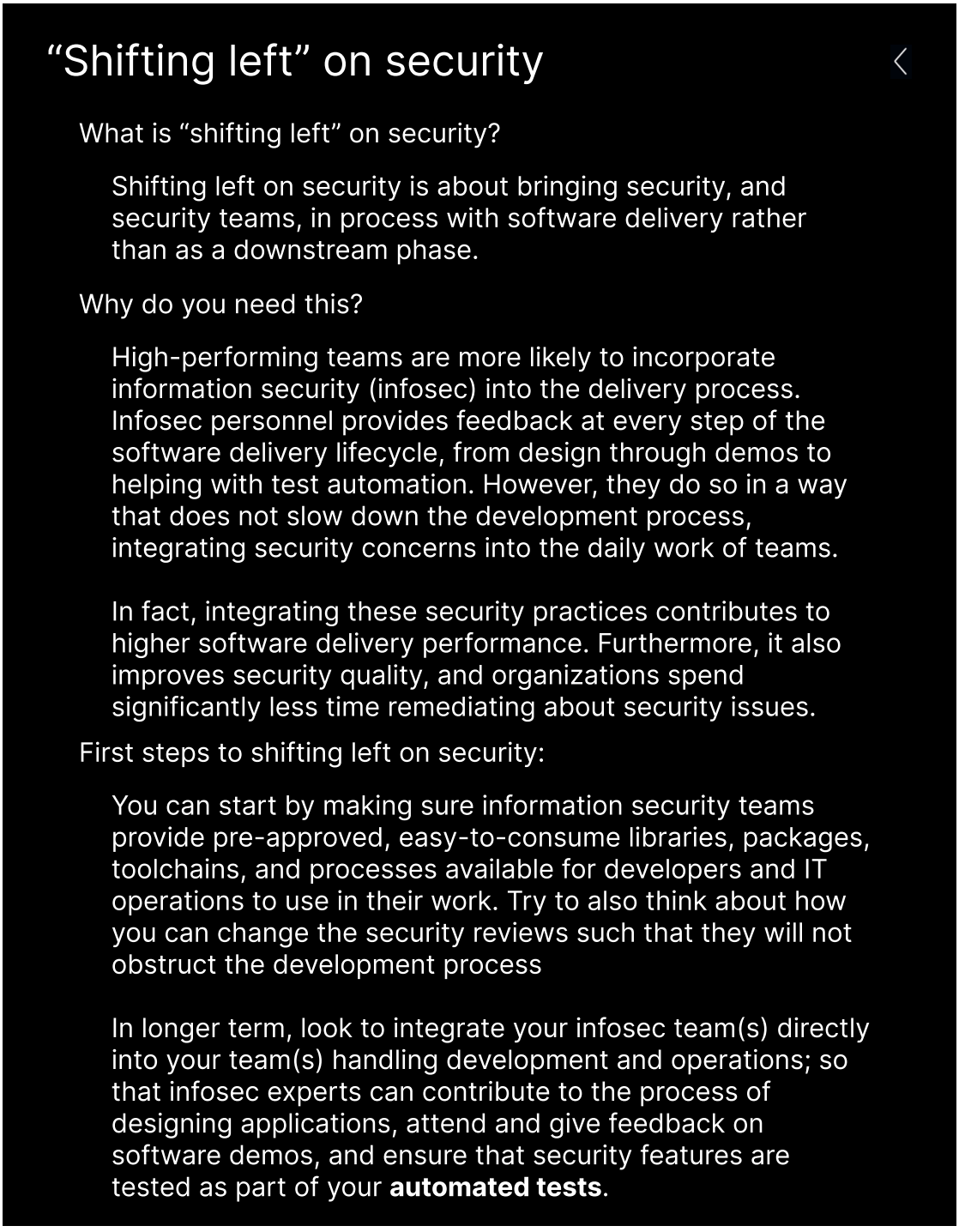

Figure B.16: Continuous Delivery capability 7 - Shifting left on security

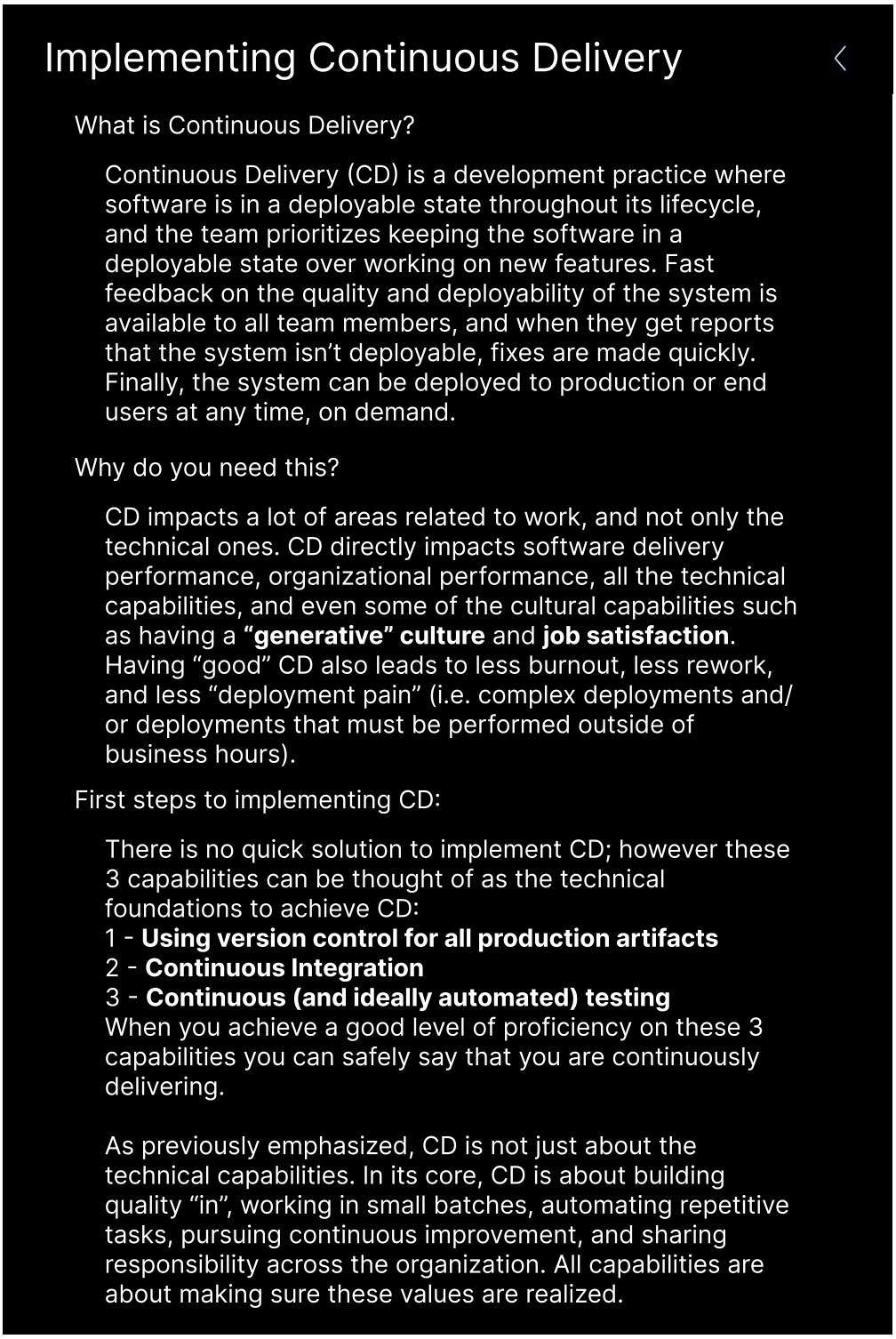

Figure B.17: Continuous Delivery capability 8 - Implementing Continuous Delivery

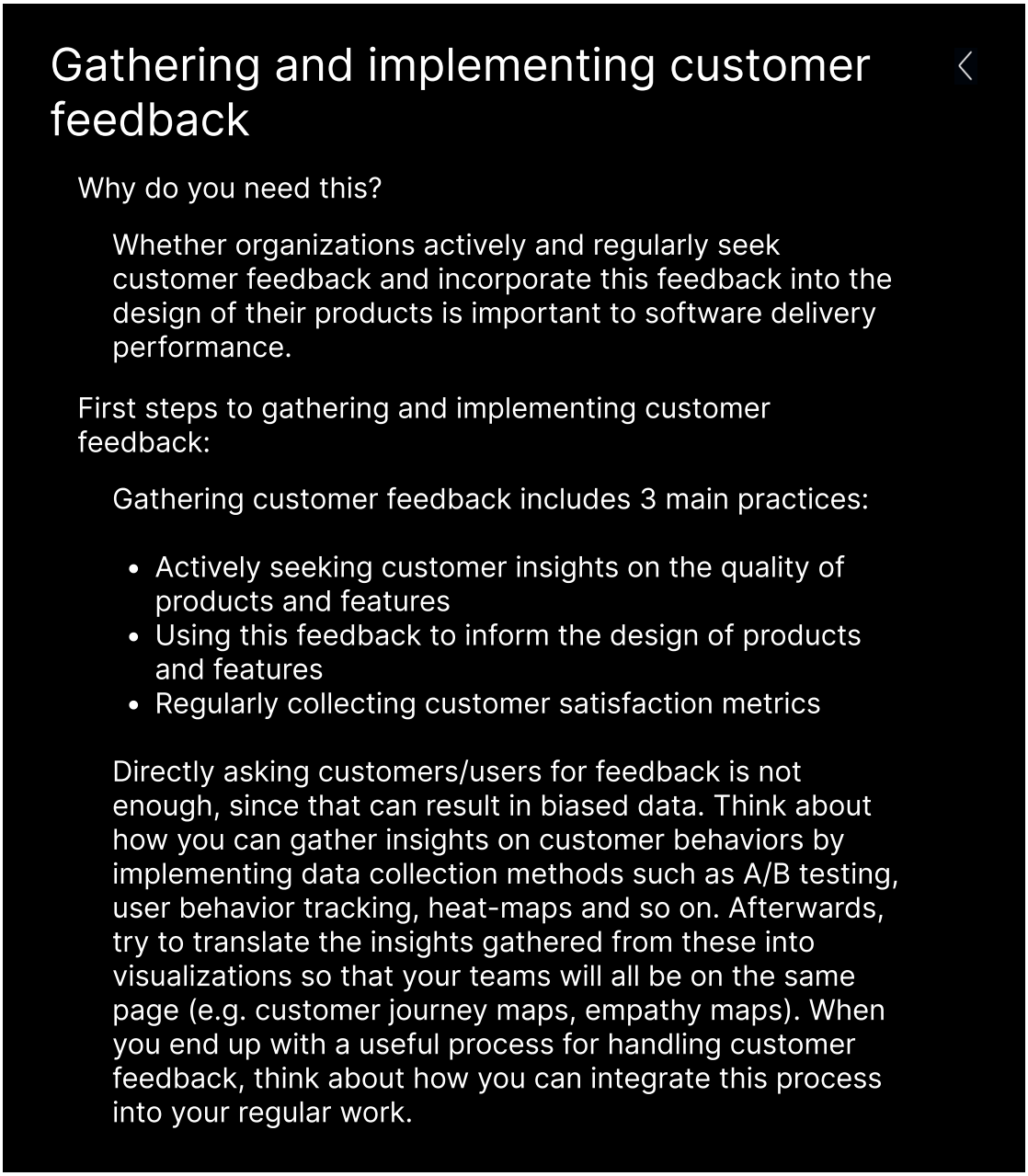

Figure B.18: Continuous Delivery capability 9 - Gathering and implementing customer feedback

Figure B.19: Continuous Delivery capability 10 - Working in small batches

Figure B.20: List of Architectural capabilities

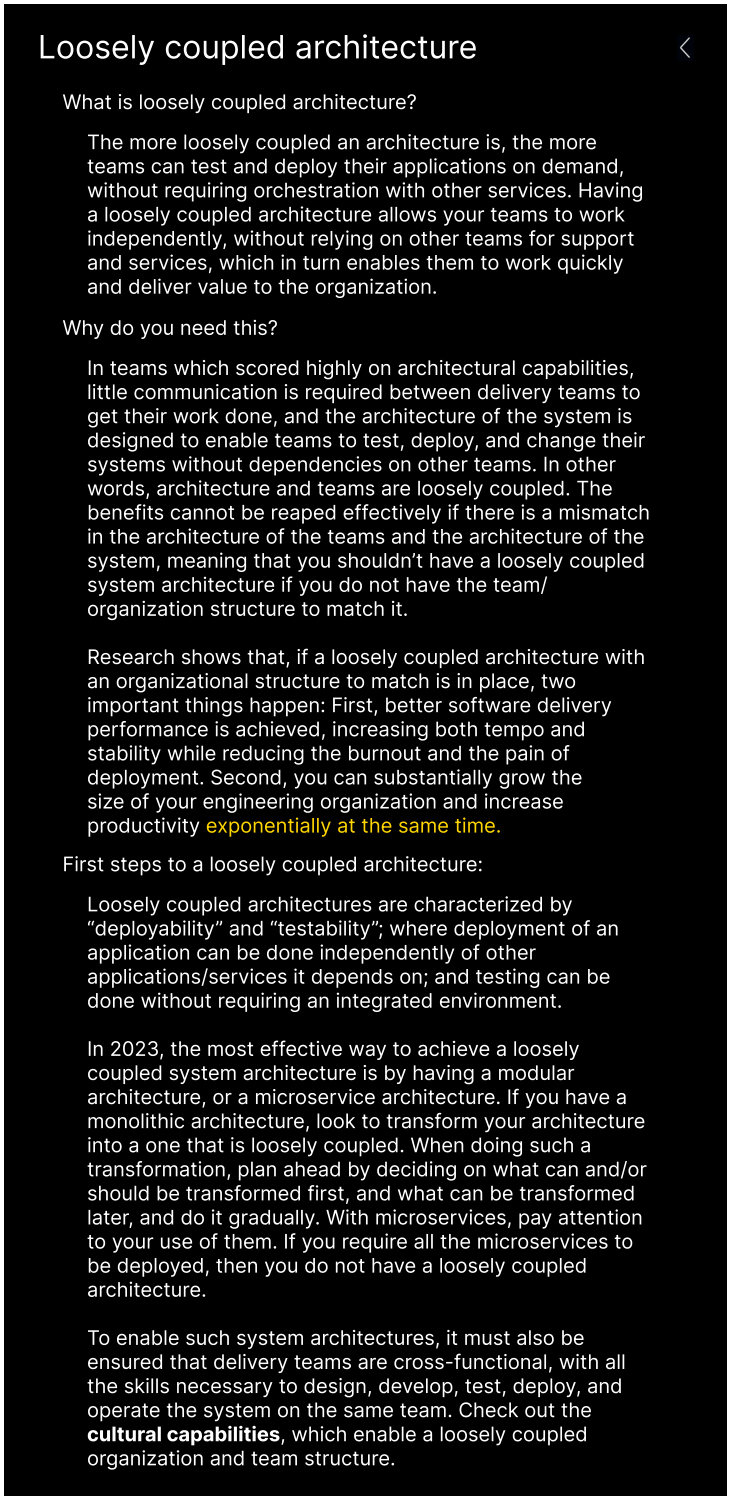

Figure B.21: Architecture capability 1 - Loosely coupled architecture

Exponential Scaling

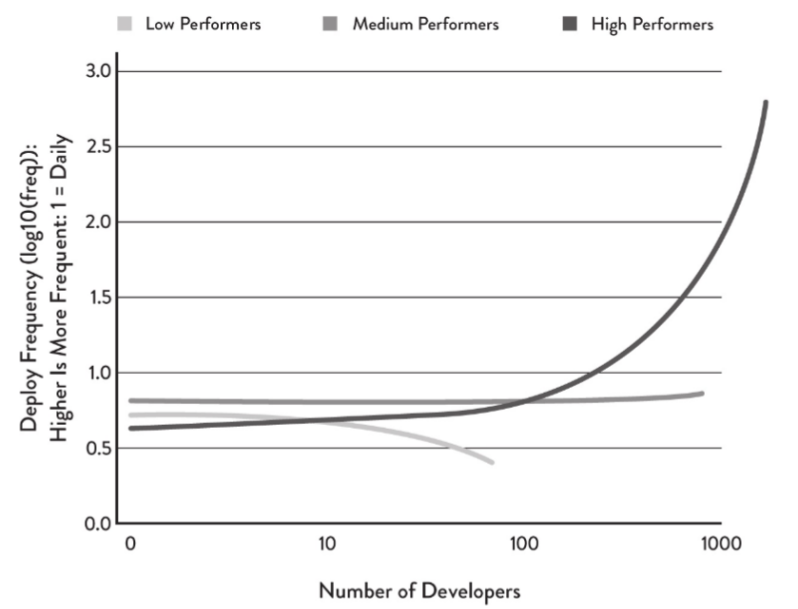

Figure 5.1: Deploys per Developer per Day

The traditional view of scaling software development teams states that while adding developers to a team may increase overall productivity, individual developer productivity will in fact decrease due to communication and integration overheads. However, when looking at number of deploys per day per developer for respondents who deploy at least once per day, research shows that:

- Low performers deploy with decreasing frequency.
- Medium performers deploy at a constant frequency.
- High performers deploy at a exponentially increasing frequency.

Figure B.22: Exponential scaling description

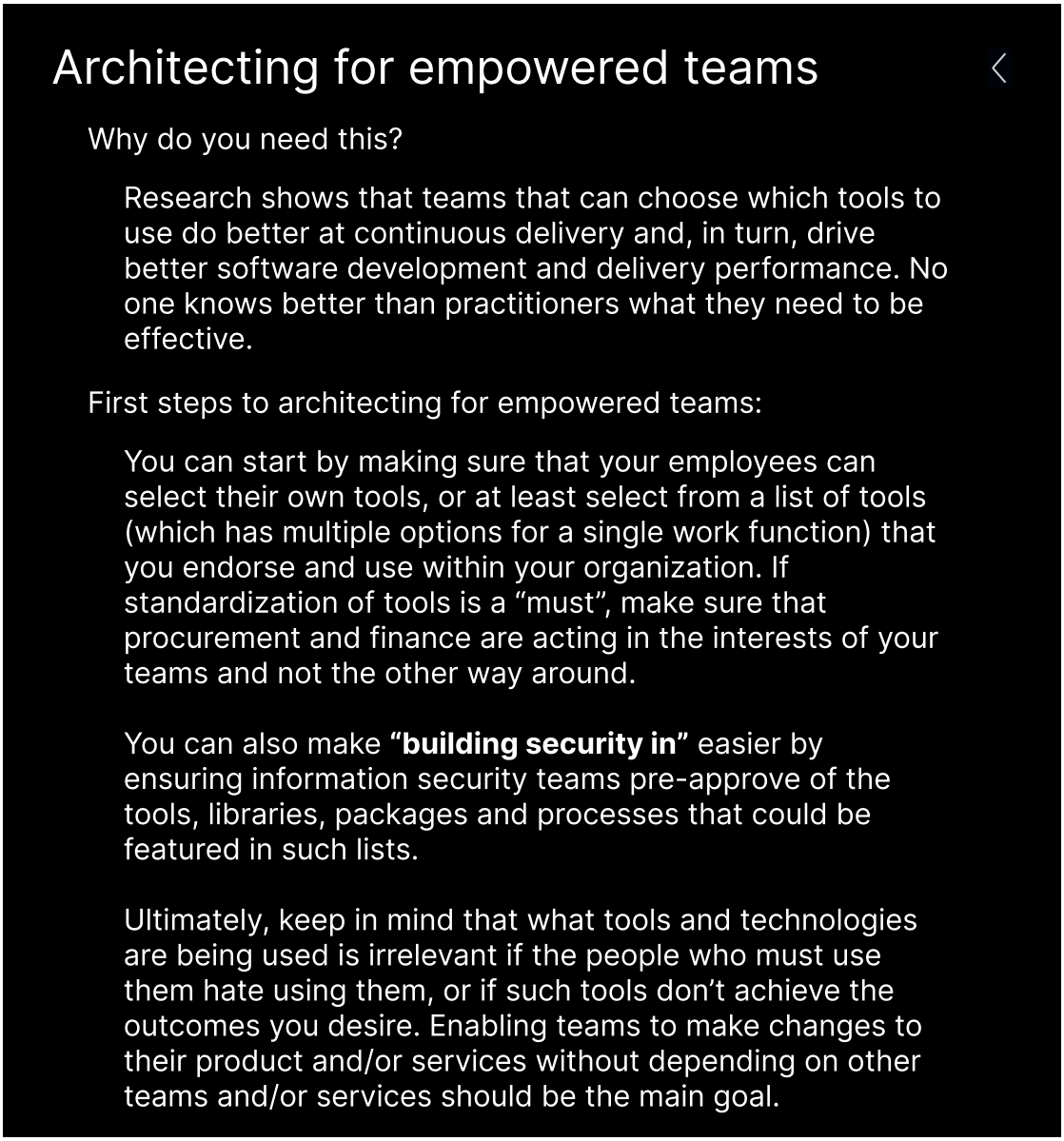

Figure B.23: Architecture capability 2 - Architecting for empowered teams

Figure B.24: List of Lean Management and Monitoring capabilities

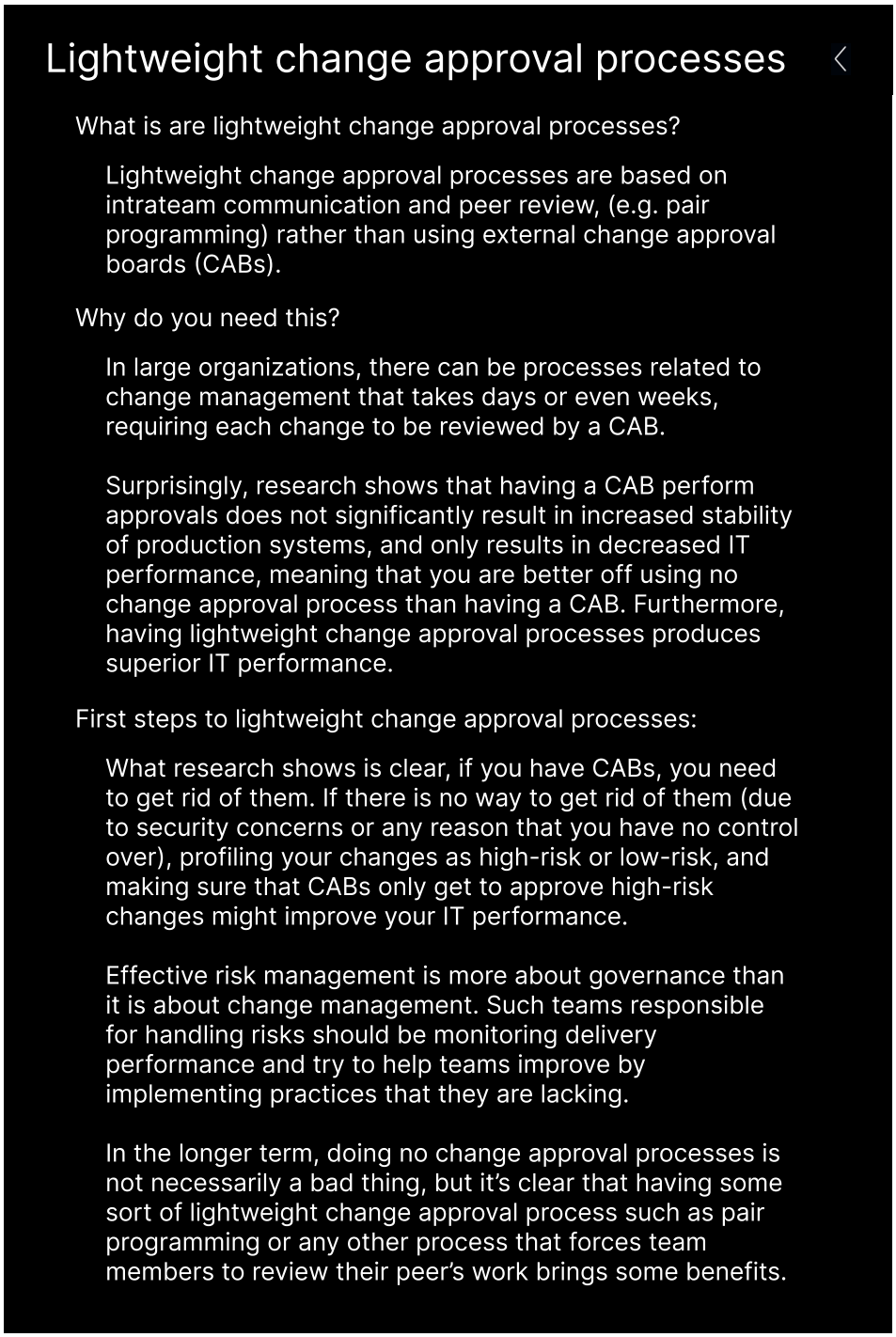

Figure B.25: Lean Management and Monitoring capability 1 - Lightweight change approval processes

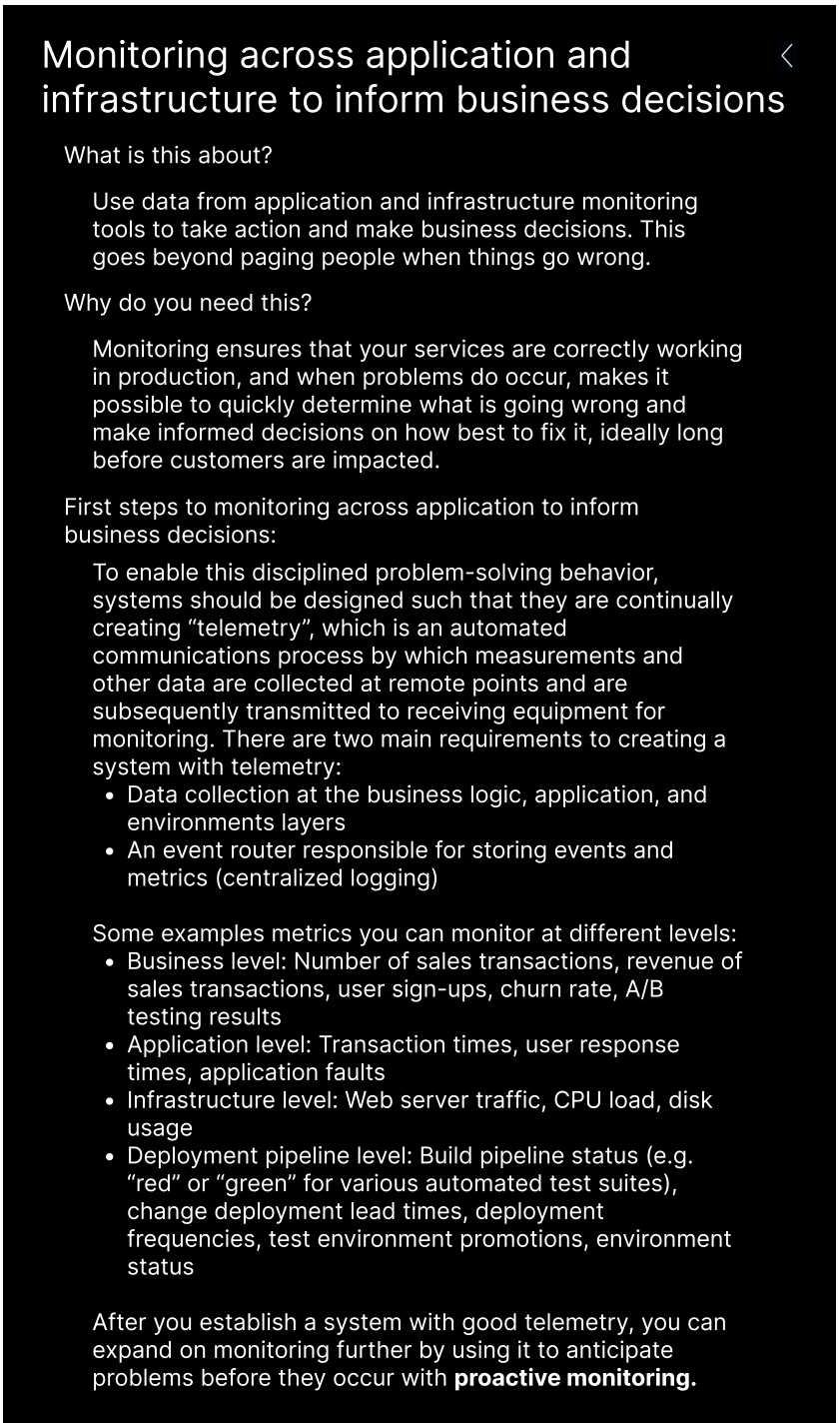

Figure B.26: Lean Management and Monitoring capability 2 - Monitoring across application and infrastructure to inform business decisions

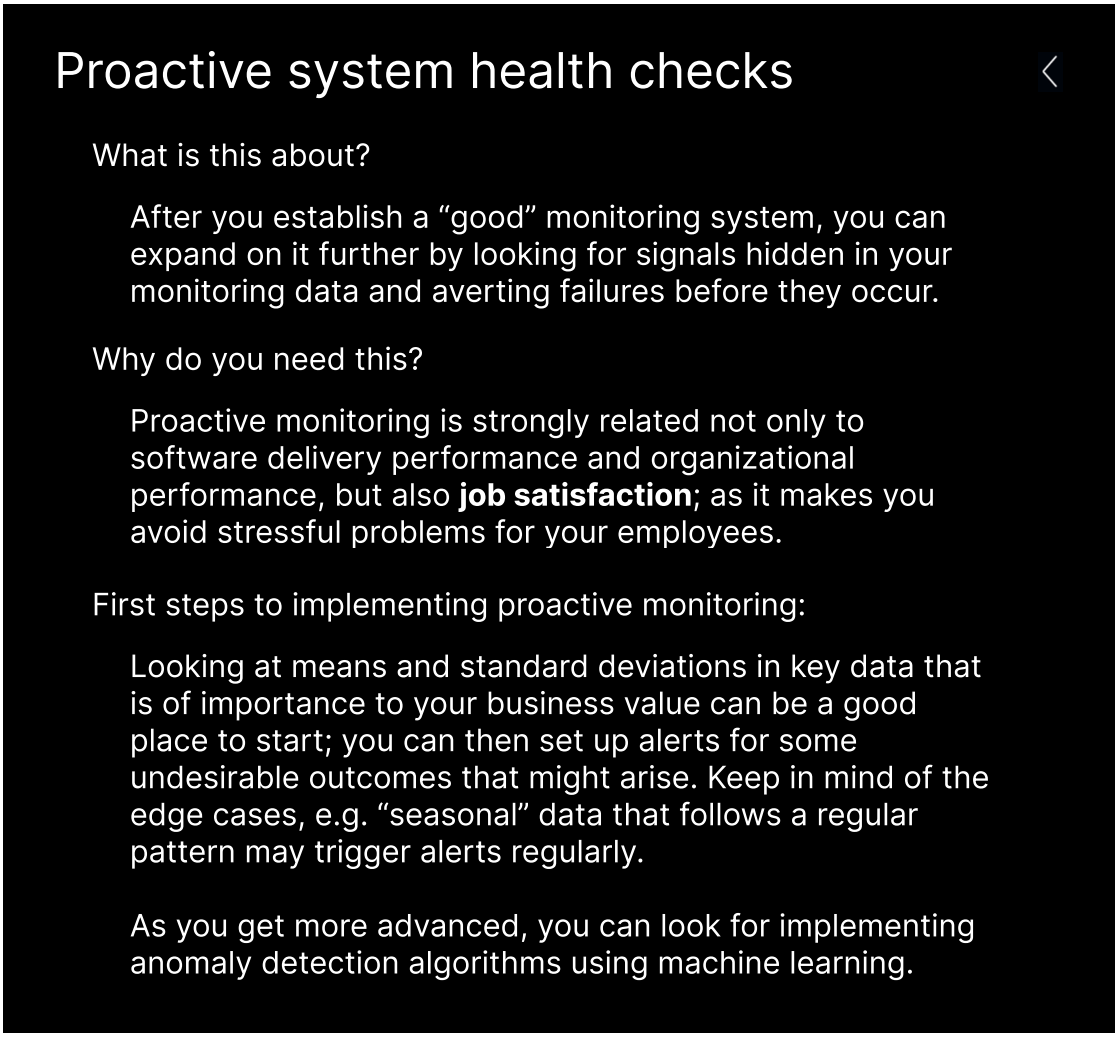

Figure B.27: Lean Management and Monitoring capability 3 - Proactive monitoring

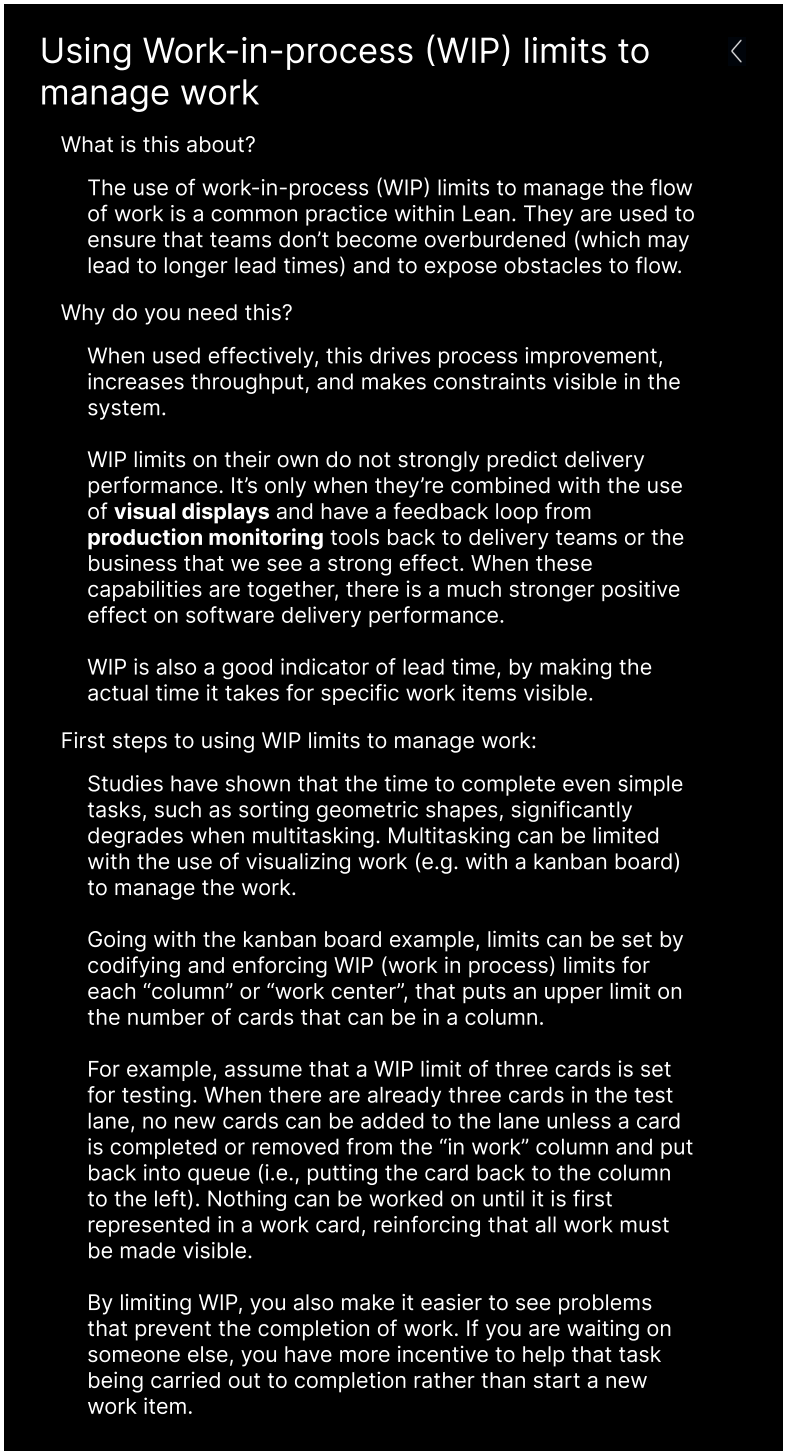

Figure B.28: Lean Management and Monitoring capability 4 - Using WIP limits to manage work

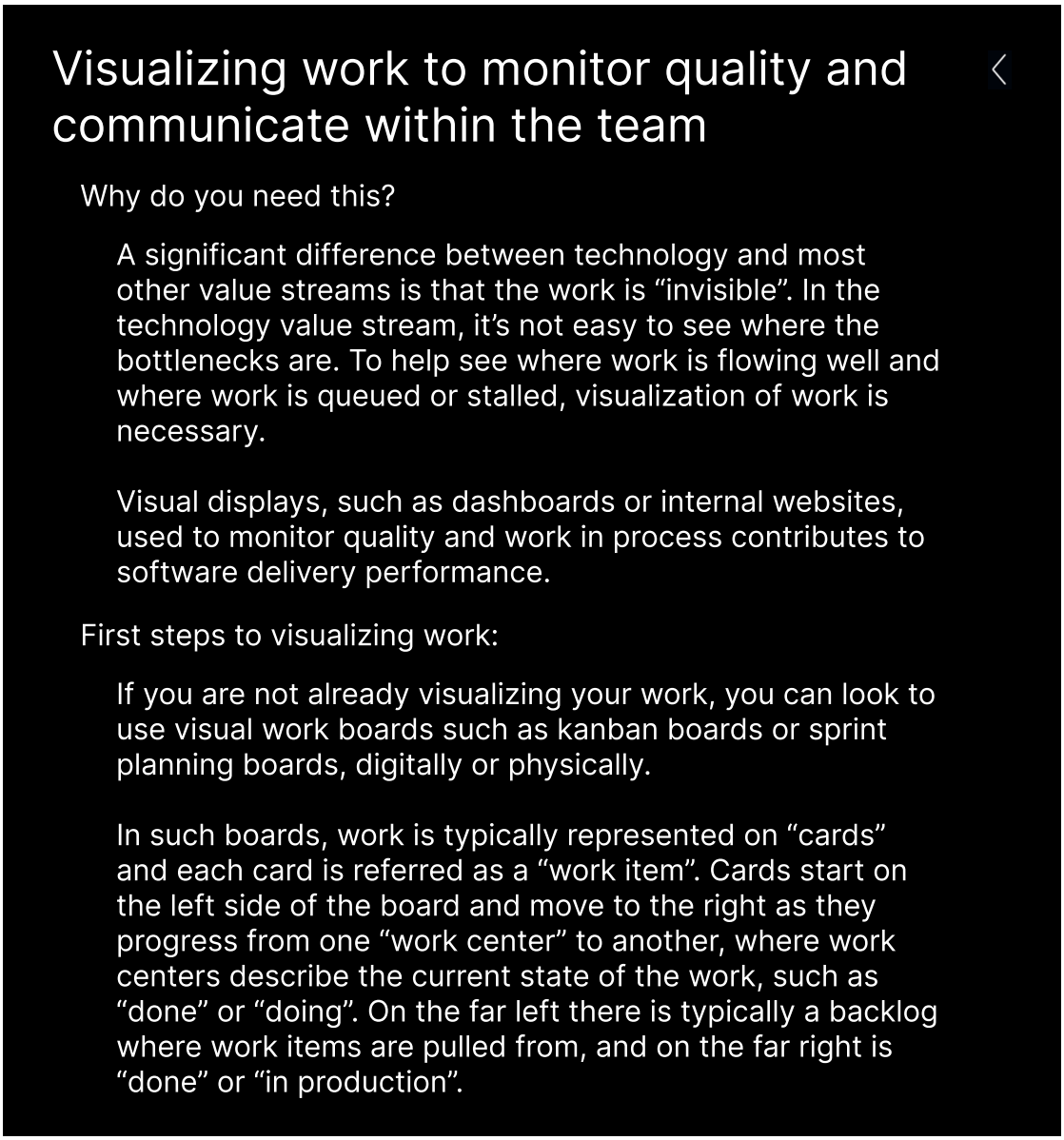

Figure B.29: Lean Management and Monitoring capability 5 - Visualizing work to monitor quality and communicate within the team

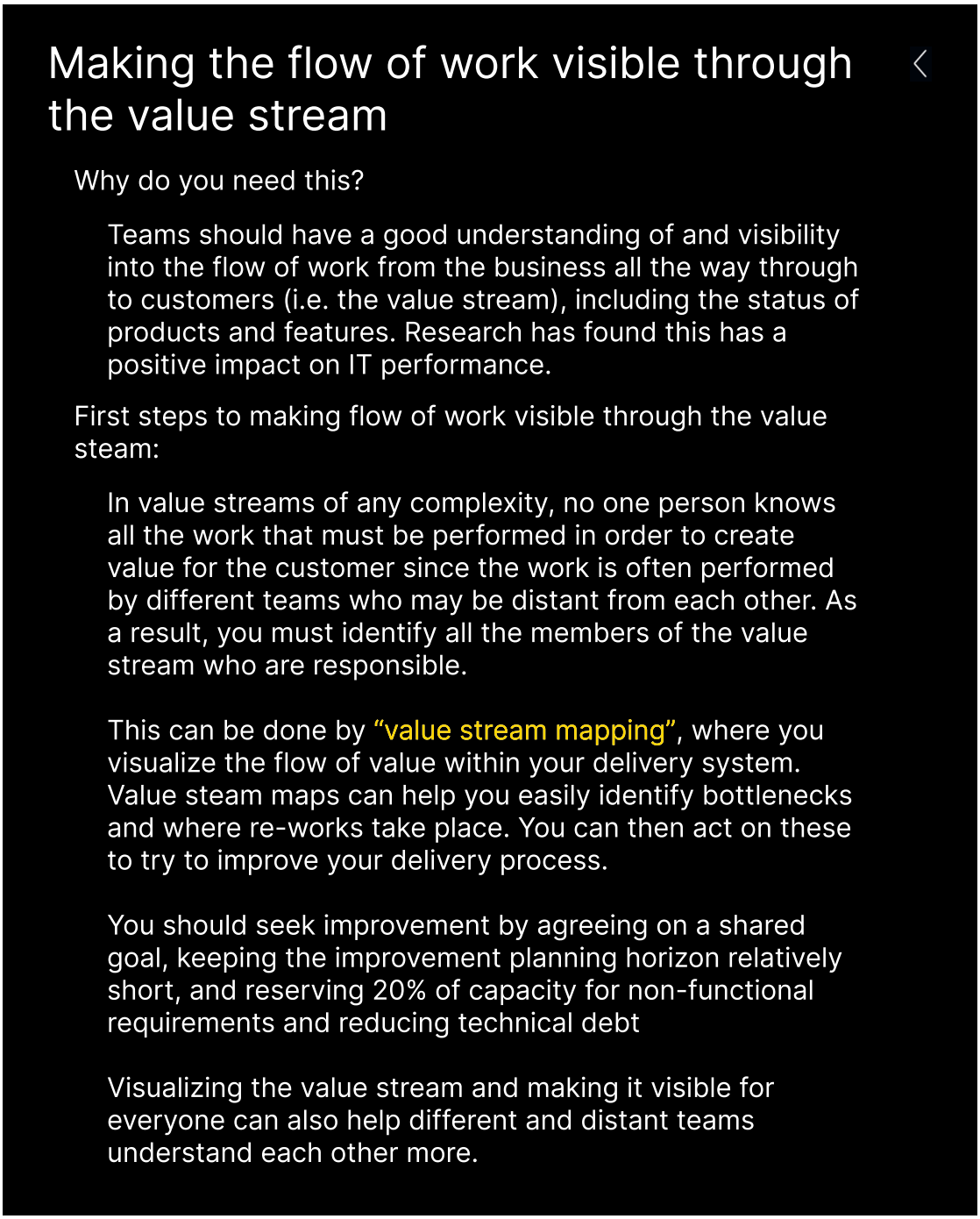

Figure B.30: Lean Management and Monitoring capability 6 - Making the flow of work visible through the value stream

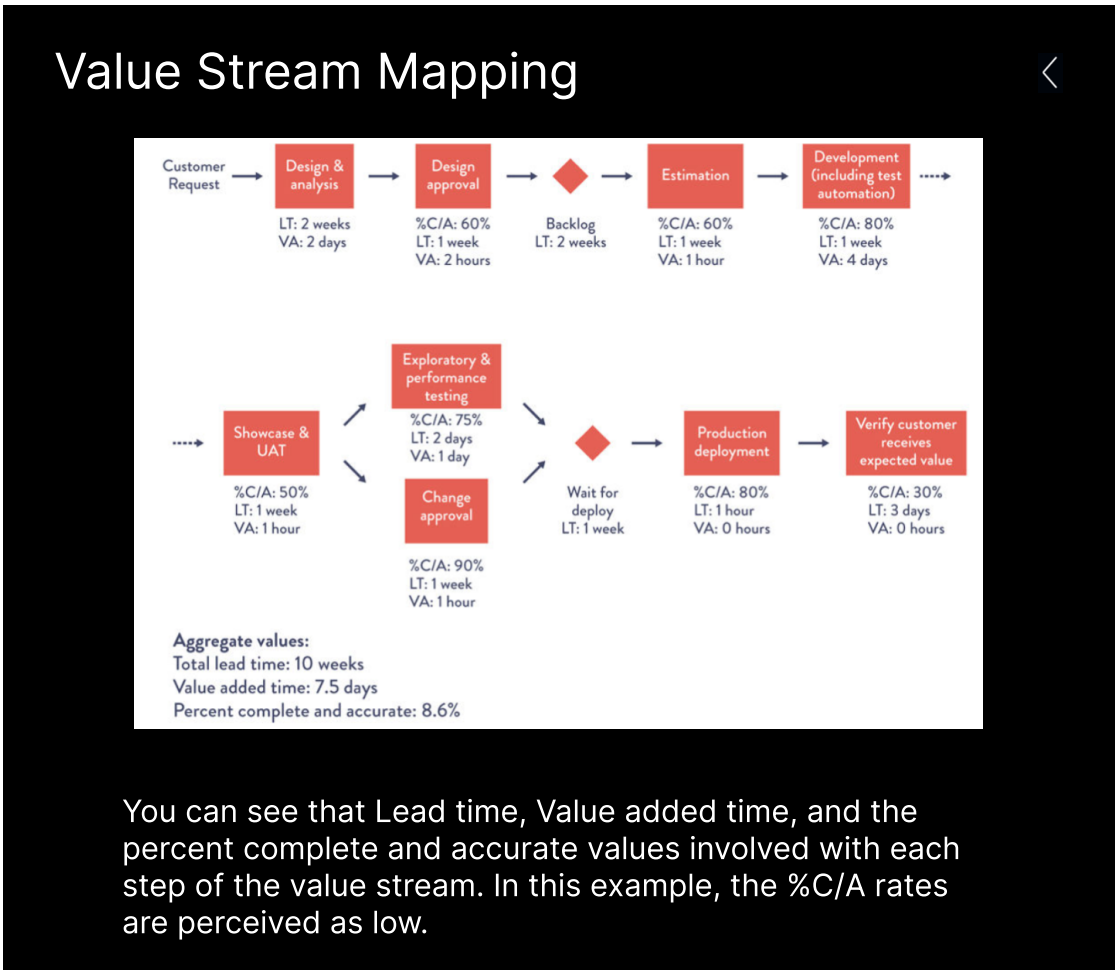

Figure B.31: Value stream mapping description