2019

Virtualization Technologies for Enterprise Agility

MSc Thesis v1.0

Tom Osinga
t.h.m.osinga@students.uu.nl
tom.osinga@pwc.com
5975336

Utrecht University
Graduate School of Natural Sciences
Department of Information and Computing Sciences
Master Business Informatics

Supervisor of PwC:
Kees-Jan Kraaier
kees.kraaier@pwc.com

Supervisor of UU:
Luz Marcela Ruiz Carmona
m.ruiz@uu.nl

PRICEWATERHOUSECOOPERS | Thomas R. Malthusstraat 5, 1066 JR, Amsterdam

This page is intentionally left blanc

Executive summary

Organizations are nowadays working on agile transformations in order to anticipate on all emerging trends in their environment. Technology is involved with almost any change in an organization. Especially in business transformation and -digitalization, new technologies are introduced. Kaddoumi and Watfa (2016) argue that businesses facing changing environments require their business and IT executives to act fast and quick towards uncertainty and unpredictability, in order to be able to accept and adapt to such changes.

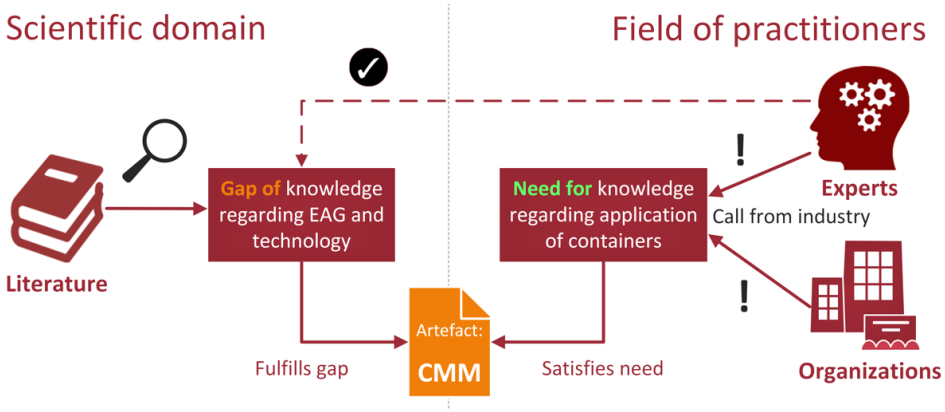
We conducted research on enterprise agility and its relation to technology. We found characteristics of enterprise agility and discovered these are applicable to enterprise components of Organization, People, and Technology (OPT). Further research on these components show dedicated research lines and specific agility sub-types for O and P. However, none of that was found regarding the enterprise component of Technology. We soley discovered several authors that denoted the importance of technology and its usage. This identified a gap of knowledge regarding the literature of enterprise agility and technology.

The domain of virtualization is widely-known as it enabled organizations to handle their infrastructure in an agile manner. Containerization is a new emerging virtualization technology that has the potential to support organizations in becoming more agile in respect to business and IT. However, containers are not indicated as being able to support an organization in becoming agile, nor the literature consists of a source that links containers with enterprise agility. This denoted the same gap of knowledge, whereas containers form the technology.

To investigate these knowledge gaps, we used the Design Science research method of Wieringa (2014). Starting with the Problem Investigation phase, we further explored the literature to highlight the gap of knowledge, and subsequently rationalize the need for conducting research in this domain. By the means of a literature review, we concluded that we found no sources that explicitly link (enterprise) agility towards technology. To emphasize this conclusion, we additionally performed expert interviews to state the opinion of the field of practice and gather further knowledge on the concepts.

As a result, the experts recognized our discovered gap, and argued that organizations have a need for knowledge regarding the utilization of containers. Hence, we strengthened the gap of knowledge and identified the call from industry. The experts additionally denoted that most traditional organization (i.e. not technology-driven) are currently not aware of the benefits of containers for their operations, or lack the knowledge on either how to start with containers, or how to improve their current container utilization. Therefore, organizations are in need for tangible knowledge on utilizing containers with varying levels of competence. Moreover, the experts emphasized that especially the SDLC process of organizations can be improved by such tangible knowledge. Referring back to the literature findings, we also found that SDLC belongs to most important use cases for containers.

Other challenges we found around containers are the security aspect, and finding the optimal architecture configuration to improve containerized architecture performance. However, we believed that developing a solution for providing knowledge around container utilization would be the most valuable. By providing differentiating levels of skill regarding container utilization, we contribute to the field of practice by satisfying its need. In addition, we contribute to the scientific domain by stating an explicit relationship between enterprise agility and technology. Therefore, we decided to design a Container Maturity Model (CMM) with the focus on the SDLC process. Ultimately, to enhance enterprise agility from a technology perspective.



The research project continued with the second phase of Design Science: Treatment Design. We defined a null (H_0) and alternate (H_1) hypothesis that describe the effect of containers in the SDLC process on enterprise agility regarding business and IT. To test these hypotheses, we executed different types of expert interviews through four iterations. We chose the form of multiple iterations for designing the CMM, in order to collect different types of information. Each iteration had a different setting and goal. In the first iteration, we performed brainstorm sessions and used the main results of the prior PI phase to define a first concept of the CMM. The remaining iterations consisted of semi-structured expert interviews with internal PwC experts, and external experts from a large international bank. This collection of varying information acted as our main source to design the CMM from. Hence, all iterations combined depict a design process of incremental nature.

We intertwined the design process with validation sessions, in order to gather additional knowledge on SDLC and perceive feedback of the experts regarding our CMM we presented. Hence, each iteration used a different CMM concept version to present the experts, which resulted in a further developed CMM version.
The expert interviews of iteration 2 and 3 are recorded. We analyzed these interview recordings by using NVivo 12. First, we defined a taxonomy with a classification of topics and detailed aspects, which we focus on during the analysis of the interviews. Our focus was to analyze the feedback where we could distil improvements from. We used the taxonomy to 'tag' periods of time in the audio files.

The result of the iterations is the designed CMM with the focus on SDLC. Here, we mainly focus on the software development aspect, and partially incorporated the software delivery aspect (pipeline) into the model. As the CMM consists of differentiating levels of skill regarding the utilization of containers, organizations can either derive knowledge on how to start utilizing containers, or use the model to indicate their current performance and state their maturity level, in order to use the metric specifications to determine how to further improve their SDLC configuration. The corresponding PDD provides an overview of the model's sequence, and supports future research by clarifying what deliverables are realized by which phase and activity.

Finally, through the PI phase, we found that before containers can be used to their full potential, they require certain characteristics in the organization's architecture, applications, and provisioning technique. Hence, we conducted research on finding such a manner of integration. We concluded that the microservices architecture pattern meets the use-case specific container requisites, as this pattern decouples applications and systems into small, independent services. This makes the applications and services short-lived, stateless, and lightweight, leading to the enablement of container advantages in the enterprise architecture. Besides that, applying the microservices pattern in enterprise architecture additionally realizes the Infrastructure-as-Code principle, which makes infrastructure programmatically. This enables even faster infrastructure deployments.

Therefore, adopting the microservices architecture pattern in an organization's enterprise architecture, enables the organization to exploit container advantages, use the principle of Infrastructure-as-code, which support enterprise agility improvement regarding business and IT. Moreover, combining these architectural principles with our designed CMM, an organization can improve its container utilization regarding their SDLC process, which applies throughout the whole organization. Hence, contributing to the overall enterprise agility regarding business and IT.

Abbreviation list

This section describes all textual abbreviations used in this document. Green rows denotes the most important abbreviations.

Concept	Meaning
AM	Agile Manufacturing
API	Application Programming Interface
CET	Cloud Enabling Technologies
CICD	Continuous Integration & Continuous Delivery
CMM	Container Maturity Model
CRZ	Containerization
EA	Enterprise Architecture
EAG	Enterprise Agility
IaaS/PaaS/SaaS " 'aaS " is used in text when referred to all concepts together.	Infrastructure-as-a-Service Platform-as-a-Service Software-as-a-Service
IaC	Infrastructure-as-Code
IT(x). This can be either **IT1**, **IT2**, **IT3**, or **IT4**.	Iteration (x).
LoS	Line of Service
MSA	Microservices architecture
OPT	The combination of Organization, People, and Technology
OS	Operating System
PI	Problem Investigation (phase according to Design Science).
RQ	Research question
SD	Software development
SDLC	Software development Lifecycle
TA	Technology Agility
TD	Treatment Design (phase according to Design Science).
VLAN	Virtual Local Area Network
VLZ	Virtualization
VM	Virtual Machine
VPN	Virtual Private Network
WoW	Way of Working

Table of contents

1. Introduction

This section elaborates on the research of this MSc thesis project. First, a context of the perceived issues is given in the problem statement. Subsequently, a research method is selected and rationalized, including a research goal, research question and corresponding sub-questions. Then, the overall research context is given, including positioning of research and practical project matters. Finally, the section is closed by descriptions of protocols for the literature review and expert interviews.

1.1. Problem statement

The problem statement describes the current trends and phenomena that are experienced in the practitioners' domain. This section aims to provide an understanding explanation to answer what the current perceived problem is, and why this situation contains potential for research. The three main subjects of this project are shown in Figure 1. This figure depicts an overview of the relationships between these subjects. Throughout this MSc thesis, these subjects are described and linked to each other. Next sub-sections describe the potential synergy between all three concepts.

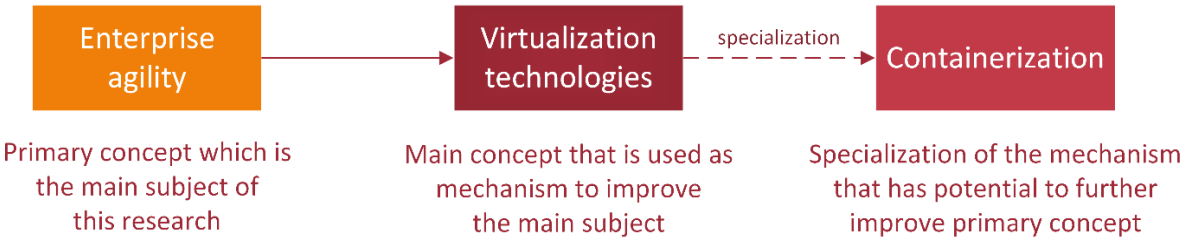


Figure 1: Relationship between three main concepts

1.1.1. Agile and enterprises

Organizations nowadays have to become increasingly more agile to anticipate on all emerging trends in their environment. Change is motivated from different needs that are difficult to predict. Factors such as hyper-competition, increasing demands from customers, regulatory changes, and technological advancements make it an important determinant of firm success (Overby, Bharadwaj & Sambamurthy, 2006). Kaddoumi and Watfa (2016) argue that businesses facing changing environments require their business and IT executives to act fast and quick towards uncertainty and unpredictability, in order to be able to accept and adapt to such changes. To successfully deal with unpredictable, dynamic, and constantly changing environments, many different solutions have been proposed, such as networking, reengineering, modular organizations, virtual corporations, flexible manufacturing, etc. (Sherehiy, Karwowski & Layer, 2007).

However, agility concepts are not yet explicitly defined and conceptualized, and there is no commonly accepted definition of enterprise agility in the scientific domain (Sherehiy et al., 2007). Nonetheless, core agility attributes have been identified, which are flexibility, responsiveness, speed, culture of change, and integration and low complexity. These core attributes can be translated into specific indices for each of the main enterprise components (Sherehiy et al., 2007). The same authors state that in reference to agility, these main enterprise components are Organization, People, and Technology (OPT).

Research towards agility for different domains exists and is applied in practice (section 3.3 and 3.4). Examples of such domains are manufacturing and software development. However, specific research on agility regarding technology is not found during the literature review. Although, what is found are studies that define the importance of technology for enterprise agility. According to Yusuf, Sarhadi, and Gunasekaran (1999), related attributes and practices of an agile organization concerning the domain of technology are 1) technology awareness, 2) leadership in use of current technology, and 3) skill and knowledge enhancing technologies. In addition, Tseng and Lin (2011) have developed a conceptual model of enterprise agility that encompasses of agility drivers, capabilities, and providers. When referring to technology, they state that *Technological innovations* is one of the main agility drivers, and *information integration (infrastructure)* is one of the important agility providers. Regarding infrastructure, Pal and Pantaleo (2005) state that organizations should configure adaptive infrastructures. Concluding, these sources describe the importance of being aware of new

technological trends and being able to skillfully exploit these new trends by having an adaptive infrastructure. Considering the rapidness of current developments in technology, the role of technology in enterprise agility is bound to increase. Based on these findings, it is therefore implied that for organizations, it is key to discover new possibilities of this increasing role of technology, becoming an expert, and apply it in practice.

Referring back to agility and its most important enterprise components. Examples for agility regarding Organization and People are known. Respectively, agile enhancements in organizational processes, and the project technique SCRUM to handle workforce. From an enterprise architecture (EA) perspective, the layers of business and application are provided with agile enabling means. The remaining layer – technology – consists of infrastructure. Different examples exist that implicitly result in agile benefits for this layer. However, a technology that explicitly enabled agility in infrastructure is virtualization. Virtualization made it possible for organizations to virtually partition their infrastructure, enabling them to use the same infrastructure components for multiple goals instead of a single goal. By doing this, multiple soft- and hardware instances can be launched on a single infrastructure component, in order to provision an organization's operations, which require computing resources or services. One sees virtualization as an agile enabling technology for technology.

1.1.2. Virtualization technologies

Currently, virtualization technologies are becoming more widely used. Whereas in the late 1960s different technology organizations were experimenting with sharing computer resources among large groups of users. Nowadays, large data centers exist where multiple virtualization technologies are making abstractions of physical hardware, and create aggregated pools of logical resources consisting of CPUs, memory, disks, file storage, applications and networking. Subsequently, those resources are offered to users or customers in the form of agile, scalable, consolidated Virtual Machines (VMs) (Oracle, 2012). Hence, virtualization enabled organizations to manage their infrastructure in an agile manner. Around 2008, virtualization technologies already realized a shift in the IT market. Moreover, virtualization was predicted to change the use of IT, pricing and licensing of software, realize new forms of applications, change the domain of operating systems (OS), and create a competition between vendors that did not exist before (Dawson & Bittman, 2008).

An example of such new vendor is Docker. This organization anticipated on the possibilities of virtualization by further developing an advanced virtualization technology, realizing a new concept of containers. This popularized another domain: containerization. In short, Docker containers consist of an executable package for pieces of software, which means that all required files and components are available in the container to be able to run on the users' (Host) OS (Linux and Windows based), and easily create a virtual environment. Compared to conventional virtualization solutions – meaning VMs – containers are launched faster and cost fewer resources, which increases scalability potential. An interesting fact is that these container characteristics align even more with the aforementioned agility attributes than conventional virtualization solutions. Figure 2 visualizes this statement about containers and their potential to support agility.

The field of practitioners noticed this alignment, which can be seen in the growth of popularity of containers in the technology domain. Since Docker's launch in 2013, the organization has grown tremendously and became a market standard for containers. Currently, Docker is spreading its technology across the IT sector. This made other IT organizations join the exploration of containerization, and resulted in the development of different container solutions. Despite the increased popularity of containers, no scientific sources were found during the literature review that state an explicit link between containers and agility.

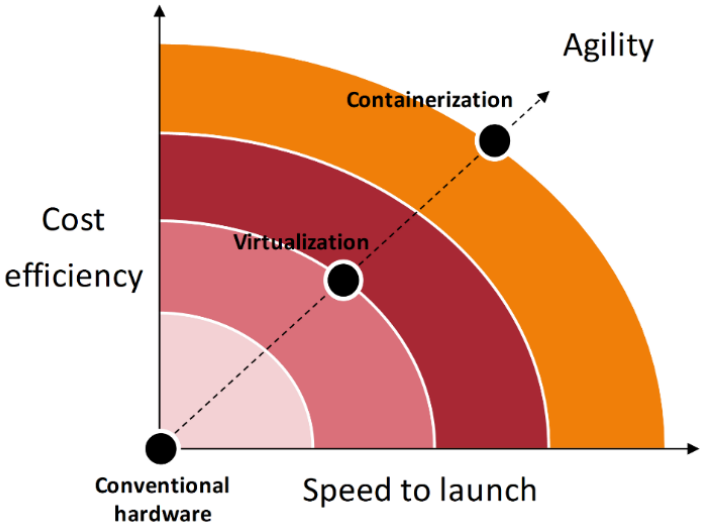


Figure 2: Containers and VMs towards agility

1.1.3. Young virtualization concept: Containerization

Due to the success of the application of different containerization solutions, the interest of containerization in the Cloud-Enabling Technologies (CET) market is increasing. Evidence to support this statement originates from different sources. According to the *2017 Annual Container Adoption Survey* of Portworx, container usage in organizations spending $500,000 or more a year on license and usage fees for container technologies, has increased to 32% up from a reported five percent in 2016 (Portworx, 2017). Moreover, organizations who are spending more than one million dollars on license and usage fees on container solutions, have been increased to ten percent, up from four percent (Portworx, 2017). The characteristics of organizations who participated in this survey vary across different industries, organization size, and the number of vendors used for different container solutions. Of these solutions, Docker is the most frequently used container solution. However, the same survey also showed that organizations are using more than one container solution in their daily operations. Another vendor, Google's Kubernetes, dominates the container orchestration (i.e. management of containers) market. Although, more vendors are still emerging, all with their own solution (technique) for containers or container orchestration. The reason for the usage of multiple solutions is because containerization is still relatively infancy. The technology is continuously under development as each vendor develops different types of functionality in their container solution, resulting in new functionalities.

Another study goes beyond percentages. 451 Research[1] conducted research on the application of container in the CET market. The study provides actual numbers of current revenue of all container technologies, and additionally predicts containers' revenue for each year until 2020. The research company states that the containers revenue was $495 million in 2015, and already increased with 123,64% to $1,107 million in 2017. Moreover, 451 Research predicts that in 2020, containers revenue will approximately be $2,2688 million (Buckley, 2017). Figure 3 depicts the graph containing the numbers per year.

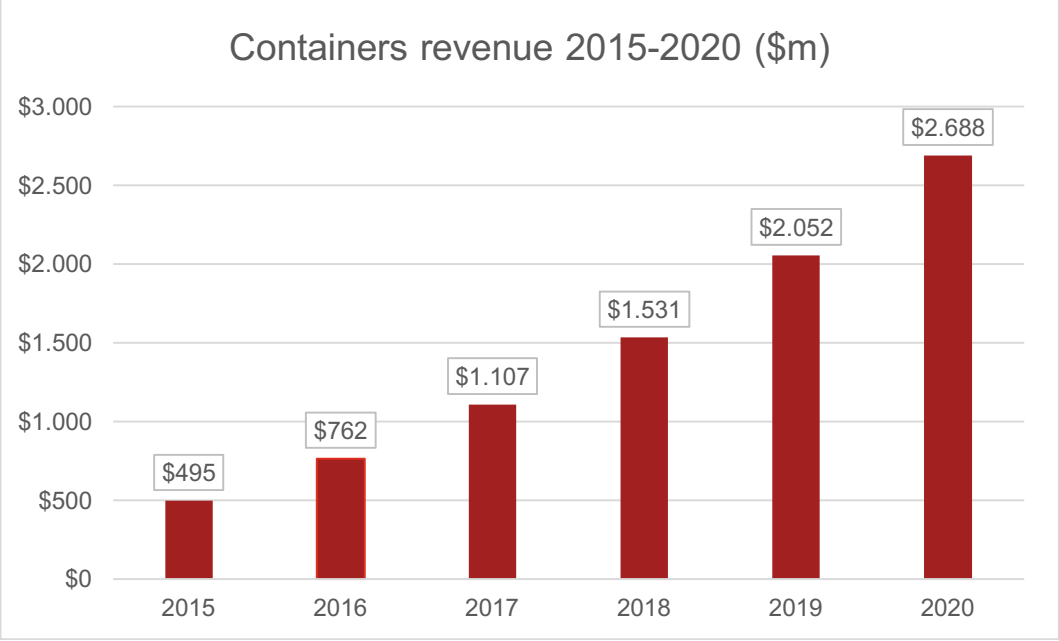


Figure 3: Containers revenue – Buckley, 451 Research (2017) (recreated)

The numbers of Figure 3 denote the popularity of containers throughout the practitioners' domains. Due to the continuous research and development onto containers, the technology will extend its functionality. This means that containers can increase their scope of application. Based on the results of the Portworx survey (2017), containers are most of the times used in agile software development environments. When applied throughout the whole software development pipeline, containers can support disciplines as Continuous Integration and Delivery.

[1] 451 Research is a research center that analyzes new technologies, services, and companies that disrupt and evolve IT. The center has more than 100 analysts that conduct research to 13 different IT domains (e.g. Cloud, security, and infrastructure).

1.1.4. Main conclusion

The role of IT is increased to a primary function, instead of being a secondary supporting function. Moreover, IT is integrated in other functions of organizations, implying the importance of efficient management of IT. Enterprise architecture (EA) provides insights about an organization's overall cohesion and dependencies of all (architectural) elements, including managing organizational aspects, application landscape, and infrastructure. This architectural information supports organizations to (re)position and efficiently manage IT elements, aiming for an architecture that facilitates agility.

As aforementioned, virtualization technologies enabled organizations to increase the utilization of their data centers. Compared to VMs, containers bring equivalent advantages, although faster, require fewer resources, and through different functionality. Hence, the benefits of containers have a stronger alignment with the core agility attributes[2] of Sherehiy et al. (2007). This alignment is shown in the next Table 1, whereas container benefits are linked to the impacted characteristics of enterprise agility in perspective of an organization's IT landscape. We excluded 'Culture of change' (see Appendix XI). Red colored virtualization functionalities are variants of these characteristics that we perceive as agile. Appendix XI provides explanations about the made connections between the EAG core attributes and virtualization technologies (VMs and containers).

Table 1: Core and global EAG attributes aligned with virtualization functionalities

EAG core attributes (Sherehiy et al., 2007)	Virtualization functionalities (Firesmith, 2017, Flow.ci, 2016; Showell, 2015; Amaral et al., 2015; Pahl, 2015; Kaur, 2018; Lewis & Fowler, 2014; IT2Int#4; IT3bInt#6)	Virtual Machines	Containers
Flexibility	1.1: *Launch of virtual environment*	√	√
	1.2: *Rapid, infra-agnostic launch of virtual environment*	.	√
	3.1: *Full application scalability support (monolithic)*	√	√
	3.2: Service-independent scalability support (microservices)	.	√
	4.1: *Full OS image usage (resource heavy)*	√	.
	4.2: *Lightweight OS images usage (resource efficient)*	.	√
	6: *Purposeful stateless application support*	.	√
	7: *Customized application runtime environment packages*	.	√
Speed	1.1: *Launch of virtual environment*	√	√
	1.2: *Rapid, infra-agnostic launch of virtual environment*	.	√
	2.1: *Automated virtualized hardware deployment*	√	.
	2.2: *Rationalized automated hardware deployment*	.	√
	4.1: *Full OS images (resource heavy)*	√	.
	4.2: *Lightweight OS images (resource efficient)*	.	√
	5.1: Multiple, complex infrastructure files and scripts	√	.
	5.2: *Easy to configure scripts*	.	√
Responsiveness	3.1: *Full application scalability support (monolithic)*	√	√
	3.2: Service-independent scalability support (microservices)	.	√
	2.1: *Automated virtualized hardware deployment*	√	√
	2.2: *Rationalized automated hardware deployment*	.	√
Integration and Low complexity	*5.1: Multiple, complex infrastructure files and scripts*	√	.
	5.2: *Easy to configure scripts*	.	√
	6: *Purposeful stateless application support*	.	√
	7: *Independent execution, communication and maintenance of (parts of) applications (microservices architecture enabled)*	.	√

[2] It should denoted that in the article of Sherehiy et al. (2007), the authors made a distinction between main agility attributes (7), and *core* agility attributes (5). The five agility attributes are denoted as 'core' as they are applicable to all aspects of the enterprise (including OPT components).

There is, however, no explicit link established between these container benefits or characteristics and agility. IT experts from the practitioner's domain are aware of container's benefits, but no sources are found that scientifically define containerization as a technology that can be used to pursue enterprise agility.

When such a link is established between agility and containers, organizations that aim to become enterprise agile can use this link to discover to what extent containers are applicable for their goal. Subsequently, EA should be applied to state the overall cohesion and dependencies of the organization. After that, the containers can be correctly positioned in the organization, providing the agile enabling characteristics of containers. However, organizations without sufficient in-house knowledge about containers are still questioning how to optimally use containers in order to become more agile.

Reconsidering the discussion about enterprise agility, virtualization, containerization and its characteristics, EA, and their interrelationship, the following problem statement is defined:

Organizations nowadays have to be more agile to anticipate on emerging trends in their environment, in order to create a proper response in time. When increasing their enterprise agility, a newly designed strategy can be executed more quickly by the whole organization as the desired response. Virtualization was one of the first technologies that enabled organizations to manage their infrastructure in an agile manner. Containerization is a young IT concept that possesses characteristics that align with core agility attributes. When compared to virtualization, containers have increased benefits in managing infrastructure and applications in an agile way. Currently, this concept is primarily used by advanced technology-driven organizations to improve agility of software development and testing. However, it is unknown how containerization can support the whole organization for improving its enterprise agility, how containerization can be integrated into EA), and how traditional organizations can benefit from using containers.

Citation/definition 1: Problem statement conclusion

1.1.5. Follow-up sub-sections

The following sub-sections of section 1 describe how this research project is executed, in which environment, and the corresponding conditions.

- **Section 1.2 Research method** describes the research method we applied to further investigate the problem context and realize a solution.
- **Section 1.3 Environment of research** further elaborates on how the research is positioned in its environment. We focus on denoting how this research interoperates with its environment, i.e. our interaction with the scientific domain and field of practice.
- **Section 1.4 Research context** describes the context of this research. This means we provide a description of the organization, and our position in this organization.

1.2. Research method

1.2.1. Research goal

For this research, we chose to further explore virtualization technologies and its subtype containerization for the primary concept enterprise agility. The aim is to discover the opportunities of combining agile concepts with virtualization, by stating a relationship between both subjects. Subsequently, to use EA to position containers in an organization, to ultimately support enterprise agility from a technology perspective. Therefore, the main goal of this research is:

> *Support enterprise agility by exploring the integration of virtualization technologies in organizations.*

Citation/definition 2: Research goal

We envision that organizations can anticipate on rapidly emerging trends and changes in their environment by communicating and executing their new strategy throughout the organization. Subsequently, to link this strategy to organizational execution to maintain their organizational success. Considering the fact that containers further enable agility than conventional virtualization technologies, we hypothesize that containers can further support organizations to enhance enterprise agility. To achieve this goal, we stated the following key milestones:

- Build supportive knowledge to ensure knowledge sharing of virtualization technologies with EA;
- Investigate virtualization technologies and their integration into EAs;
- Evolve current domains of EA in terms of EA alignment and agility.

1.2.2. Research questions

Considering the given context and studied problem statement, in order to facilitate an answer for the perceived phenomena, we define the following main research question:

How can organizations support their enterprise agility regarding business and IT by integrating virtualization technologies?

The Design Science research method of dr. R. Wieringa (2014) is selected as main research method. We chose this method as it combines descriptive research with exploratory research, in order to design an artefact to improve the problem context. As we aim in this research at combining enterprise agility with virtualization technologies, this research is of investigative nature. Section 1.2.3 further elaborates on the characteristics and rationale of this method.
To develop a suitable answer for this research, we distilled the different concepts of the research question and processed these concepts into smaller sub-questions (RQs). These RQs are grouped in several main tasks, using the structure of the Design Science methodology. Subsequently, we further investigate and explore each RQ. This results in several research questions per phase of the chosen research method.

RQ1 Problem Investigation

RQ1.1: What is the state of the art on enterprise agility, virtualization, and containerization from a literature perspective?

This RQ elaborates on the knowledge domain of enterprise agility, virtualization, and containerization. For each concept, we searched definitions, possible standards and best-practices, descriptions of cases-studies, current developments, and the history of it. We complemented both virtualization and containerization with descriptions about how the technology works. In addition, we explained the difference between containers and conventional virtual machines. Our goal is to state a theoretical foundation from where of information can be used to build new knowledge.

RQ1.2: What is the current state of enterprise agility and containerization according to experts from the field of practice?

RQ1.2 elaborates on the perspective of experts regarding the topics of RQ1.1. With this RQ, we attempt to test the results of RQ1.1 against the practitioners' perspective. In addition, we discuss container benefits and

current challenges, as well as the role of containers in organizations and their potential to improve enterprise agility regarding business and IT. Finally, we ask the experts about their opinion concerning an artefact that uses containers to improve enterprise agility.

RQ1.3: How are enterprise agility and virtualization technologies linked to each other?
After gathering and formalizing knowledge of RQ1.1 and 1.2, the knowledge is used to link both main concepts to each other. As there is no direct link known, we expect that a deep-dive into both concepts will be made to discover and establish a link in the detailed aspects between both concepts.

RQ2 Treatment Design

RQ2.1: What are existing solutions for the topic of SDLC and maturity models?
For this RQ, we conduct research on existing methods, solutions and other related studies regarding SDLC and maturity models. Additionally, we propose to study related topics and to explore how we can process our enterprise agility and virtualization literature results towards potential maturity levels. We will use the collected information to start designing concept versions of the proposed maturity model (section 5.3).

RQ2.2: How can containerization be integrated into enterprise architecture to support enterprise agility?
We use RQ2.2 to further analyze the application of containers in practice. This includes studying literature and performing desk research. We proposed to use SDLC as our use case to focus on for applying containers. However, to integrate SDLC and its architectural elements into enterprise architecture, the supporting application and infrastructure components have to be studied as well. Therefore, we decided to study a modern architectural pattern that aligns with containerization: microservices architecture.
By doing this, we will gather knowledge to ultimately define how we can implement containers with respect to enterprise architecture. In addition, resulting knowledge can be used to support the application of the proposed container maturity model.

RQ2.3: How can we construct a maturity model regarding the implementation of containers in the software development lifecycle?
We will use the gathered information of all prior research questions to start creating concept versions of the proposed container maturity model. We consider corresponding architectural changes that are required in order to fully utilize containers. During the development of these concept versions, we will perform additional interviews with external experts from the field of practice to gather extra knowledge. Their responses support the process, as we can modify and refine the container maturity model according to their feedback. As a result, we aim to deliver a concept design that is testable in a validation session.

RQ3 Treatment Validation

RQ3.1: How do experts from the field of practice perceive the designed artefact?
We will interview multiple experts from the field of practice by combining expert interviews with validation interviews to test the concept version of our maturity model. During these sessions, we observe the interviewees' behavior during their usage of the CMM. Expert's feedback will be used to modify and further refine current concept versions of the model, realizing an improved concept version.

1.2.3. Design science

Design science is the design and investigation of artefacts in context (Wieringa, 2014). The philosophy is that artefacts that are studied are designed to interact with a problem context, in order to improve something in that context. Hence, design science problems are improvement problems (Wieringa, 2014). This means that an artefact is interacting with, and hence influencing a context. Wieringa (2014) states that it is the interaction that determines the effect of an artefact in context. An example of such interaction is given in Figure 4.

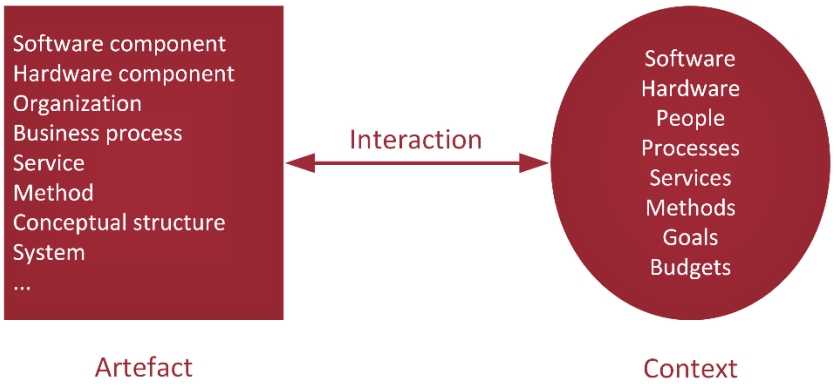


Figure 4: Design science: artefact interacting with a context - Wieringa (2014) (recreated)

In order to create a proper problem improvement artefact, the context has to be understood. To be able to understand an artefact and its context, design problems have to be investigated so that new designs can be researched on. Besides that, knowledge questions have to be answered as well. Answering these questions creates a theoretical foundation of new knowledge that is used to develop new designs. This synergy between investigating design problems and answering knowledge questions is seen as an iteration. Figure 5 depicts this iteration between the problem-solving activities.

Figure 5: Design science iteration - Wieringa (2014) (modified)

Furthermore, this research method distinguishes two main research problems: Design problems and Knowledge questions. "*Design problems call for a change in the real world and require an analysis of actual or hypothetical stakeholder goals*" (Wieringa, 2014). For instance, a solution is a design. There can be many differentiating solutions, which are evaluated by their utility with respect to the stakeholder goals. An example is of a design problem is: Design a validation method for assessing the integration of virtualization technologies.

Knowledge questions "*do not call for a change in the world but ask for knowledge about the world as it is*" (Wieringa, 2014). In contrast to design problems, it is assumed that there is solely one valid answer to knowledge questions. They are evaluated by facts and realized by empiral research. An example of a knowledge question, in reference to previous example, is: Is the assessment of the validation method explicit enough?

1.2.3.1. Design science framework

The iteration of design science is the main part of this research method. However, the cycle is accompanied by two interactive contexts: Social context, and Knowledge context. The cycle and contexts combined forms the design science framework. Figure 6 depicts the framework including all relationships.

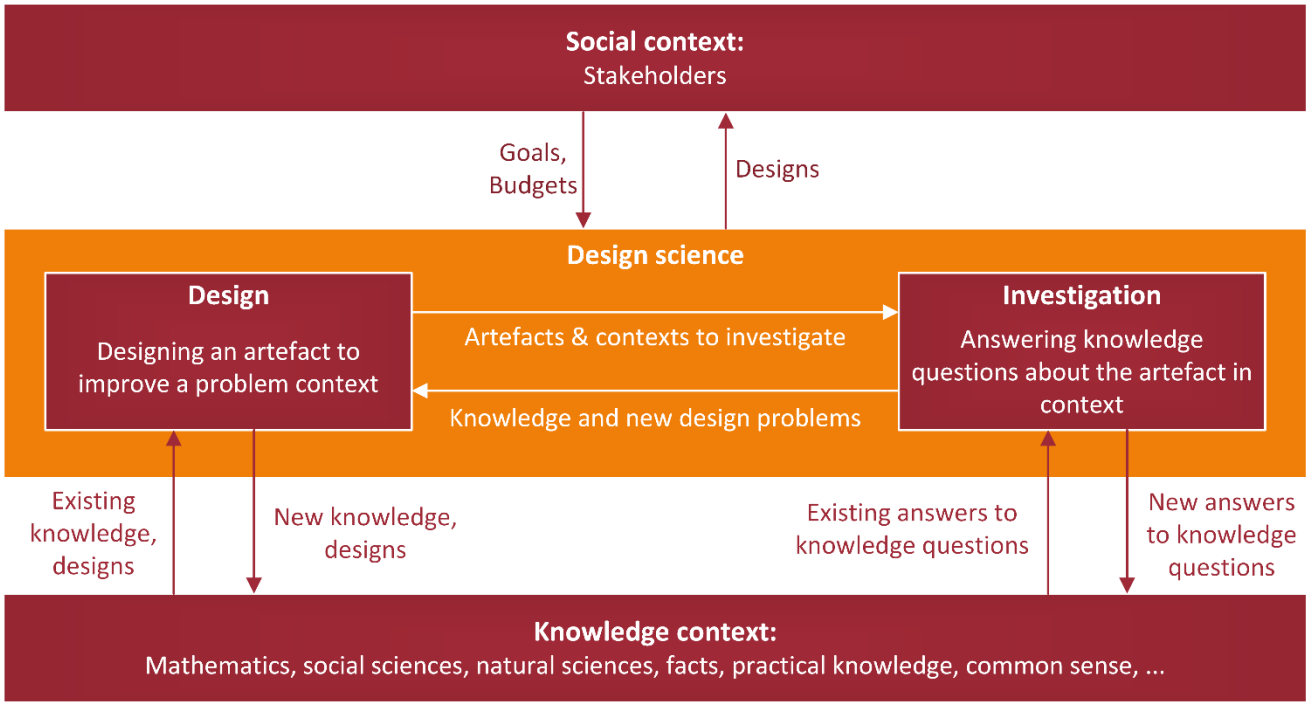


Figure 6: Design science framework - Wieringa (2014) (modified)

Social context consists of any type of stakeholder who may affect the project or may be affected by results of the project. The knowledge context contains all the existing knowledge (theories, literature, domain knowledge etc.) of all fields of science.

1.2.3.2. Design Cycle

Knowledge questions are answered by empirical research, meaning that data is collected and analyzed. In order to extract correct data from sources, this research project applies two types of research questions:

- **Descriptive questions**: solely asks for what happened (facts), without asking for explanations.
- **Explanatory questions**: ask *why* something has happened. Can be 1) causal, 2) result of production, and 3) rationalization of reasoning.
- **Exploratory questions**: to discover what will happen when combining artefacts, concepts with influencing aspects.

Design science facilitates a phased method named the Design cycle. This cycle consists of several so-called *tasks*. The activities of investigating and designing are performed in these tasks. In total, there are three main tasks:

1. **Problem Investigation**: during this task, the perceived problem is investigated. This includes identification of stakeholders and goals, denoting and studying main and additional concepts to develop a theoretical foundation, and validating findings with experts from the field of practitioners.

2. **Treatment Design**: after studying the perceived problem(s) and corresponding concepts, developing a theoretical foundation, and processing the perspective of experts, we defined the problem context and proposed a solution (artefact). Subsequently, we started to design this artefact to improve the problem context. Activities include specifying and testing requirements, study existing treatments (designs), conducting expert interviews, and using all gathered information to design and validate new treatments.

3. **Treatment validation**: in the final phase, the new design is validated by its ability to impact the problem context. Measurements are used to validate the effects of the artefact on stated requirements. Additionally, interactions with other artefacts and contexts can be evaluated as well.

These tasks are ordered using the stages of the Design Science methodology of prof. dr. Wieringa (Wieringa, 2014). The following sub-sections contain descriptions of above tasks applied in the context of this research. The sub-sections are followed by a visual representation of the applied Design cycle including all tasks.

1.2.3.3. Task 1: Problem Investigation

The first task of this research is focused on exploring the state of the art and the current phenomena in enterprise agility, virtualization technologies, and containerization. We aimed to develop a theoretical foundation where enterprise agility is described and explained, by conducting a literature review. A corresponding literature review protocol is provided within the literature review section in section 3.1. In the same framework, we described virtualization technologies and containerization, and explained how these technologies work. The combination of these concepts is used to attempt to confirm a gap of knowledge regarding the concepts. The hypothesized gap of knowledge is the lack of an explicit relationship between enterprise agility and technology. More specifically, the potential benefits of virtualization technologies for enterprise agility. In this attempt, we studied both the scientific and practitioner perspective to denote the need for a solution to improve the problem context as described in section 1.1.

In the literature review, the main concepts are followed by additional concepts. These additional concepts were required to investigate, as a consequence of the initial literature review and expert interviews findings. Section 2 Introduction of Problem Investigation further elaborates on this.

For each of the studied concepts, we studied definitions, explained milestones in corresponding research lines, denoted possible standards and best-practices, analyzed possible cases-studies and trends, and provided a brief history of each concept to show its growth over time. In addition, we performed expert interviews to test the results of the literature, as well as to perceive the need of the practitioners' domain. A corresponding expert interview protocol is given at the beginning of section 4. Ultimately, the goal is to state a theoretical foundation from where of information can used to build new knowledge.

The deliverable of Problem Investigation is the **Long Proposal**. This document includes a literature review, results of expert interviews, a PI conclusion and a subsequent proposal for designing an artefact to improve a problem context.

1.2.3.4. Task 2: Treatment Design

The second task is aimed at designing an artefact to improve the given problem context. During this phase, we plan to conduct research on the following concepts:

- Existing solutions around the topic of SDLC and maturity models;
- Integration of containerization into enterprise architecture;
- How to design a maturity model for containers and SDLC.

These concepts are studied in order to gather information for the development of an artefact. Information that is aimed for comprises existing methods and solutions about each concept, and other related literature. The gathered information is used in order to define what a successful integration of enterprise agility and virtualization is, with respect to organizations' enterprise architecture. In addition, practitioners' expertise is collected by performing expert interviews and verification interviews.

As a result, the deliverable of this task is a **draft version** of the **Master thesis**. The thesis document includes draft versions of the artefact, final presentation, and scientific report. All products combined contribute to the scientific body of knowledge and practitioner domain.

1.2.3.5. Task 3: Treatment Validation

During the third task, we validate the artefact by conducting interviews where we ask the experts about their opinion on the presented artefact. The results of these interviews lead to an indication of possible improvements that can be processed into the artefact.

Resulting in the **final version** of **Master thesis**, and all formerly included products.

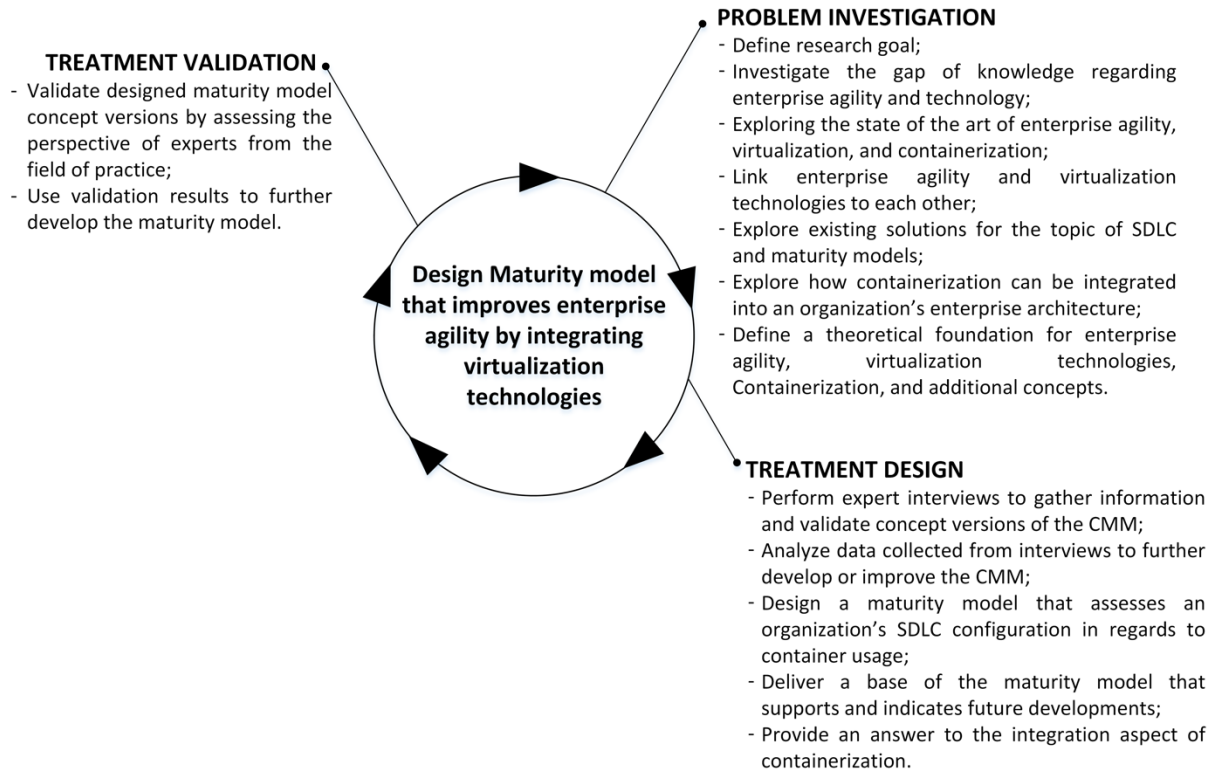


Figure 7: Design cycle - based on Wieringa (2014) (modified)

1.3. Environment of research

Section 1.3 describes the positioning of this research project in terms of the Information System Research Framework (ISRF) of Hevner, March, and Park (2004). This technique is selected to incorporate, since it enables the researchers to denote the overview of resource consumption and contribution of this research.

1.3.1. Positioning through Information System Research Framework

Design science method is aimed at developing an artefact in context to provide a solution for the given problem context. The design science cycle depicts the planning of the whole research project, including different phases and subjects of interest. The ISRF model is recreated and complemented with the defined RQs of this research project to show the RQs' consumption and contribution. Figure 8 gives this specific overview of the model.

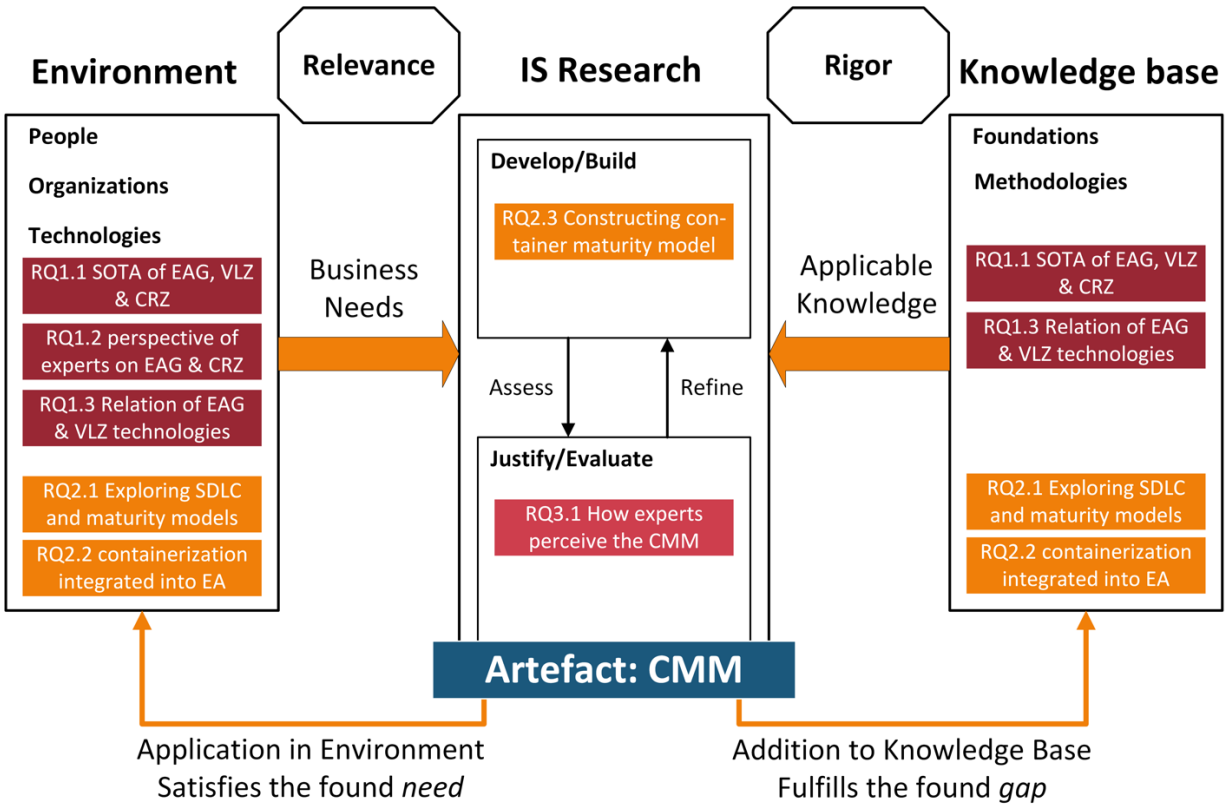


Figure 8: ISRF complemented - based on Hevner et al. (2004) (recreated)

As mentioned before in Section 1.2.2, the RQs have different goals and (re)source usage. RQ1.1 and RQ1.3 are answered by conducting a literature review and desk research. Most sources are scientific papers and originate from the Knowledge Base. In addition, business-oriented articles are used secondary to complement the theoretical foundation. RQ1.2 is answered by performing semi-structured interviews with experts from the field of practice. These expert interviews also supported the answering of RQ1.3. Therefore, both RQ1.1 and RQ1.3 are positioned in both Environment and Knowledge Base, and RQ1.2 in Environment.

To answer RQ2.1 and RQ2.2, additional knowledge is required. This is extracted by studying both literature and business-oriented articles. The remaining RQ of the TD phase are design tasks. Hence, RQ2.3 is positioned in the IS research facet. Subsequently, when the theoretical foundation is built, experts from the field of practice (Environment) are approached to gather extra knowledge. This field is also approached during the validation phase of this research, in order to investigate the expert opinion about the designed artefact. By answering the final sub-research question, the artefact is developed, and problem context is resolved.

Conducting research to the aforementioned subjects and concepts leads to a knowledge base that is used to build an artefact. This artefact is the answer to both the perceived gap of knowledge regarding enterprise agility and technology, and the need for knowledge on applying containers.

1.4. Research context

The final section of the introduction describes the context this master thesis project is executed in. At first, the organization where the project is performed is described. Secondly, the position of the student in the organization is explained. Finally, the relevance of this research for all parties is described.

1.4.1. PricewaterhouseCoopers

PricewaterhouseCoopers (PwC) is an international network of multiple accounting firms that is active in 158 countries. Originally, the network was started by Samuel Lowell Price as an accountancy firm in 1849. Throughout the following centuries, several mergers have been performed that led to an extension of the overall network and created different brand names. In 1998, Price Waterhouse merged with Coopers & Lybrand, resulting in PricewaterhouseCoopers. Finally, PricewaterhouseCoopers formally shortens its brand name to PwC, as it is currently known (PwC, n.d.).
PwC's total number of employees is above the 236,000, who all work on one of the three main lines of services:

- **Assurance**: performs audits on financial statements and focusses on checking of data and processes.
- **Tax & HRS**: supports organizations with their tax strategies, -planning, and -compliance.
- **Advisory**: provides advice on different facets of organizations. This includes strategical innovation, improvement of processes and systems, advice on deals, mergers and acquisitions, and support during crises as a consequence of breaches in cybersecurity (PwC NL, n.d.).

The lines of services are offered at many industries. Examples are Retail and Consumer, Financial Services, Technology, Media & Telecom, and the Public Sector (governmental organizations). In all of the projects, PwC aims to create value for their customers, PwC's employees, and for society. The purpose of PwC comprises the following: *to contribute to the trust of society and to solve important problems*. PwC envisions that close collaboration between all the international firms can achieve their purpose. This means that it is aimed for that all teams have multi-disciplinary competences, different expertise, employees with diverse (national) backgrounds, and that each individual can be himself or herself.

1.4.2. Position of student in PwC NL

The student is active in the Advisory Line of Service (LoS) of PwC NL. This LoS is active in different industries and contains multiple amounts of teams. The student performs his activities in the Financial Services (FS) industry, and is placed in the Technology team (FS TC).

The student takes of role as specialist (intern). In the specific context of PwC, the student is named as a specialist. In formal, this position is not seen as a full employee. However, informally the student is part of the team. During the daily operations, the student can join in projects to learn about the field of practice. Although, primary focus is on conducting research and write the required master thesis documents.

1.4.3. Challenges and opportunities

Returning to the main subjects and their interrelationships in this research project. Considering the fact containerization is a relatively immature domain, more (sub)types of the virtualization technology or new functionalities/innovations are likely to emerge. PwC is interested into exploring the different types of virtualization technologies, and to discover to what extend these technologies can be used to improve enterprise agility. Especially due to the increasing dynamic environment and rapid changes organizations need to confirm to, improving enterprise agility regarding an organizations' infrastructure and IT provisioning, while integrating virtualization technologies, is nowadays strategy to improving enterprise agility.

Therefore, this research project is of interest to both PwC and the university. PwC will increase its knowledge regarding the potential of containers in reference to supporting enterprise agility. Subsequently, the results of the research can be used to further conduct research on, or to apply in practice. The university also embraces the body of knowledge resulting from this study. The results contribute to scientific domain in stating a link between virtualization technologies and enterprise agility.

Problem Investigation

2. Introduction of Problem Investigation

The Problem investigation of Design science (Wieringa, 2008), is the first phase of this research project. As shown in the section 1.1 Problem statement, we briefly pre-investigated the described knowledge domains to find the relevance of this project. Based on the results of this pre-investigation, we determined the structure and objectives of this phase. The following research techniques are selected to use in the problem investigation:

- **Literature review** on the selected concepts;
- **Expert interviews** focused on those concepts.

At the start of the project, we had a global idea on what all the main concepts comprised of. However, initially we did not possess a significant amount of experience nor affinity with the concepts. Especially the details of the later introduced SDLC concept was considered as a new domain for us. Therefore, we chose to conduct a literature review on all concepts, in order to deep-dive into each concept and explore the current challenges, solutions, and existing knowledge to discover the scientific perspective on these concepts, and to enrich our understanding on these concepts. As our expertise is related to the chosen concepts, we considered the literature review as sufficient to comprehend the concepts. Regarding SDLC, besides literature reviewing the concept, we held off the record meetings with PwC experts to further understand the concept[3].

The results of the literature review would provide us with an understanding of the concepts, the state of the art, and related challenges. Subsequently, we wanted to approach the experts and use our collected knowledge to ask about their opinion regarding the concepts and corresponding scientific perspective. We aimed to use the combined results of the literature review and expert interviews to define a proposal which includes a to be designed artefact that would anticipate on the studied concepts.

Next sub-sections describe for each problem investigation sub-phase what our intention, objective(s), and aimed conclusions were, and into what this eventually resulted in. We described an overview of the structure of all concepts that are included in the literature review in section 3.1.1.

2.1. Problem Investigation research process

2.1.1. Literature review

As aforementioned, we chose to conduct a literature review to all relevant concepts to increase our knowledge about them. More precisely, the goal of the literature review is to define the state of the art of the three main concepts, and find information that indicates a gap of knowledge regarding enterprise agility and the application of virtualization technologies. Furthermore, enterprise agility (section 3.3 & 3.4) is studied to understand the overall concept and to denote in which aspects research is conducted and still lacking. Subsequently, virtualization technologies (section 3.6) and containerization (3.7) are studied to find out how these subjects can be linked to enterprise agility. By conducting research on the details of both virtualization subjects, it is envisioned that tangible results can be found that have potential to link the subjects to enterprise agility. However, first the gap of knowledge must be indicated. Then, this gap can be used to denote the scientific and practitioner's need of linking virtualization technologies to enterprise agility and its application.

By defining the state of the art of enterprise agility, virtualization, and containerization, we intend to answer RQ1.1. Besides that, using both the knowledge of the literature review and expert interviews, we aim to state an answer for RQ1.3 – how enterprise agility and virtualization technologies can be linked to each other.

As a result, we found the relationships between the concepts and were able to indicate the scientific gap of knowledge. The corresponding conclusion is explained in section 5.1. Besides that, by studying all these concepts, we built a theoretical foundation that is used to support the answering of the main RQ

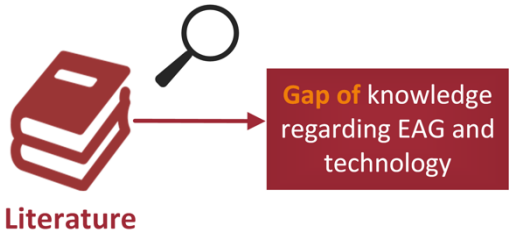


Figure 9: Literature review objective

[3] In the Treatment Design phase, we performed expert interviews whereas one of the aspects also was to further discuss the concept of SDLC and its relationship with containers. Section 6, 7, and 12 further elaborate on this.

and achieving the corresponding main research goal. Moreover, as an addition to the theoretical foundation, we further investigated other related concepts (see section 2.4). This theoretical contributes to solving the indicated knowledge 'gap' and 'need' that is experienced respectively by the scientific domain and practitioners.

2.1.2. Expert interviews

With the expert interviews, our objective was to discover the perspective of the field of practice on enterprise agility and containerization, and to validate the found gap of knowledge from the literature review. By doing this, we would establish the answer to RQ1.2. In addition, we provide ourselves with evidence that the field of practice recognizes the gap of knowledge. Moreover, this would strengthen our found gap of knowledge and further increase the relevance of this research project. Additionally, we aimed for discovering the experts' ideas on how they would fulfill the gap of knowledge with a relevant product or artefact. Subsequently, combining their suggestions with our literature findings, we intended to form the proposal to continue our research with.

Eventually, the experts gave their opinion about the found gap of knowledge and provided us with suggestions on how to fill this gap. One of the main findings is the use case of SDLC as relevant use case to apply containers in order to support enterprise agility. Section 4 elaborates on the results of the expert interviews. Finally, section 5.2 describes the conclusion of the expert interviews.

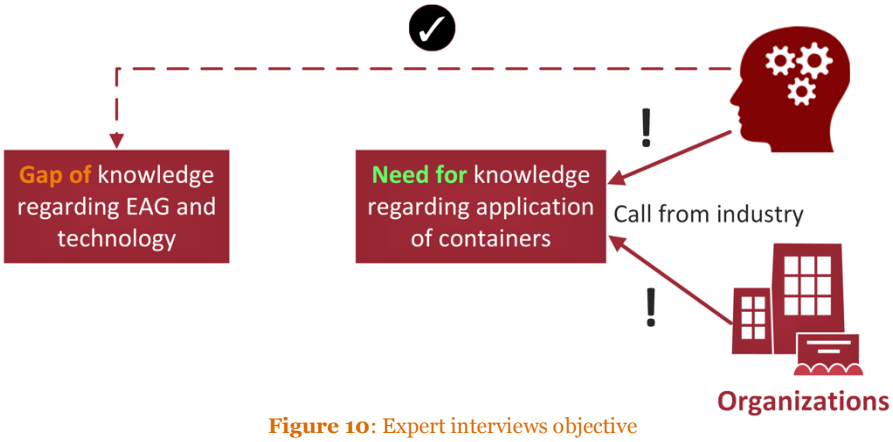


Figure 10: Expert interviews objective

2.1.3. Proposal phase

The results of both the literature review and expert interviews are combined in the proposal. Due to combining the results to each other, we aimed to establish a valid relationship that would scientifically justify the chosen artefact to design.

The chosen artefact of the Long Proposal required us to conduct additional research on the concepts of SDLC, maturity models, and CICD. Besides that, we also had to explore the possibilities on how to integrate container-related solutions into the EA of an organization. Therefore, we proposed to extend the literature review with these *additional concepts*, in order to complement the theoretical foundation and to ultimately answer the main RQ. We also found that we had to strengthen the scientific relationship between virtualization technologies and enterprise agility.

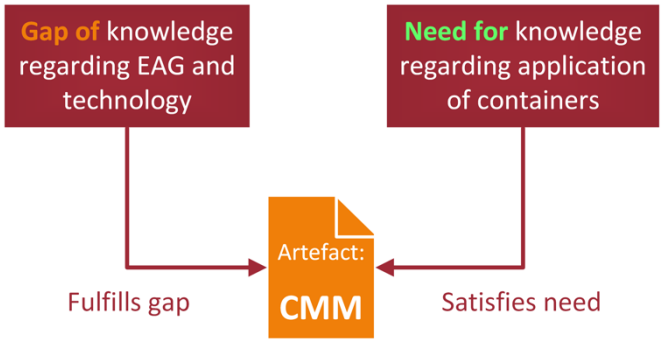


Figure 11: Result of combining literature review and expert interviews results

2.1.4. Transition to Treatment Design phase

After finishing the additional research, we finalized the theoretical foundation to start designing the proposed artefact. Subsequently, based on all PI phase findings, we created a table consisting of the most important characteristics of all related concepts that we intend to include in the proposed artefact. With this table, we continued the research process by transferring to the next phase of the research project, Treatment Design. This phase is positioned from section 6 and starts on page 79.

3. Literature review

This chapter describes the results of the literature review towards the following concepts: Agility, Enterprise agility, Technology agility, Virtualization, and Containerization. Section 3.1 describes the literature review protocol. Section 3.2 provides a preface of the literature review, including a brief explanation about its structure and the remaining sections. Both of these sections solely describe and explain review aspects of the initial concepts. This means the additional concepts are excluded from the literature review protocol and –preface descriptions, as we had a more accurate idea on what aspects of the additional concepts we wanted to review.

3.1. Literature review protocol

This section describes the protocol that is applied to perform the literature review. In this study, three main concepts are studied for the literature review: 1) Enterprise Agility, 2) Virtualization, and 3) Containerization. Table 2 shows these main concepts and related concepts. The concepts additionally define the scope of this research. All concepts are used for the vast majority in search engines, mostly Google Scholar, to gather different types of scientific sources. As these concepts are also widely used within commercial organizations, desk research consists of relevant knowledge as well. This means that conventional Google is additionally used to collect sources about these concepts. As a result, a knowledge base with both scientific and practical (business) perspectives on the same concepts is created. These sources include:

- Scientific articles and journals;
- White- and business papers;
- Reports and other non-scientific (web) documentation that originates from the field of practitioners.

Table 2 shows the main concepts of this research. Per concept, synonyms and related concepts, including linkages to other concepts are defined to indicate the spectrum of relevant concepts and their positioning.

Table 2: main concepts and their details

Concept	Synonyms or related concepts	Linkage to other concepts
Enterprise Agility	Agility	Enterprise architecture
	Organizational agility	IT alignment
	Agile manufacturing	Strategic alignment
	Workforce agility	
Virtualization	Virtual machines	Virt. vendors: Xen, Hyper-V, KVM
	Time-sharing	Multiple OSs
	Virtual runtime environment	SaaS, PaaS, IaaS
	Cloud computing	
Containerization	Container orchestration/ management	Microservices (architecture)
	OS-level virtualization	Service Oriented Architecture
	Containerized architecture	IT architecture
	LXC containers/Docker containers	Continuous Delivery/Integration

The first round of papers was collected by using concepts of Table 2. The papers (n=69) are structured in an Excel sheet that includes a column per article that indicates: title, author(s), keywords, year, and other remarks. The abstract and keywords of each paper are used to determine its relevance, and are graded by the means of a Likert scale (1=highly irrelevant – 5=highly relevant). Subsequently, papers ranked five (5) were studied at first. Information found was used to write sections of the theoretical framework. We also included references to other relevant papers we found during this analysis. This resulted in extra sources in the literature review we would otherwise have missed, and thereby complementing the overall knowledge on the concepts. After that, papers ranked four (4) were studied to find more information regarding each concept. After that, we decided per section whether the content was sufficient. Papers ranked as three (3) and lower were read if a section was determined as insufficient. Figure 12 provides a visualization of this research process. Additionally, the figure depicts how it connects to the remaining research process for the development of the Long Proposal.

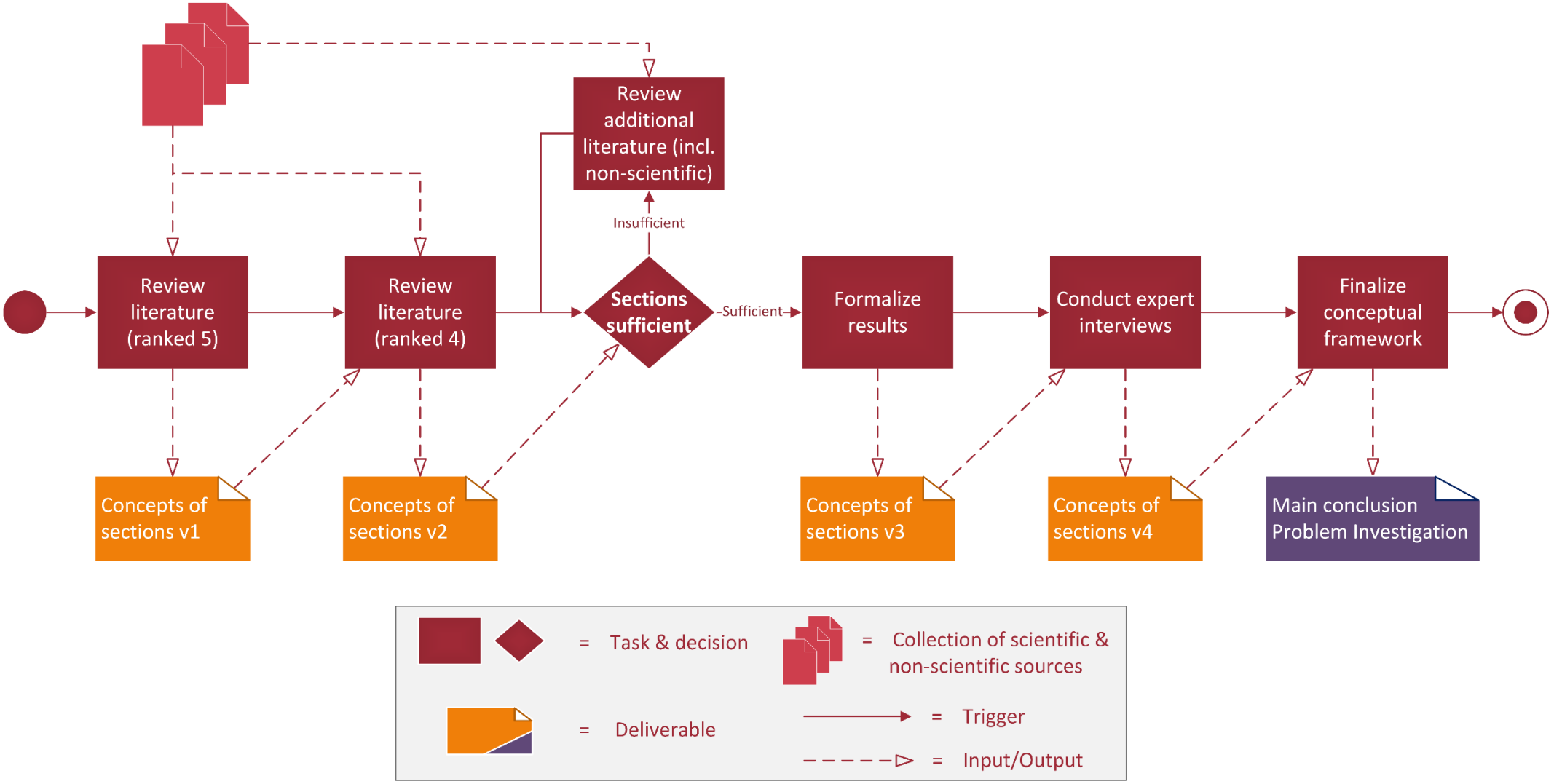


Figure 12: Problem Investigation research process

3.1.1. Literature review structure

As aforementioned in the introduction of Problem Investigation (section 2.1), we complemented the literature review with investigations to additional concepts. To provide an overview of the remaining structure of the literature review, we stated its structure below, depicting each sub-section accompanied with a brief description.

- **Section 3.2 Preface of literature review** provides the reader with context about agility, denoting the starting location of the literature review, and explains how the literature review evolves towards the other concepts. This section has sole focus on the concepts of section 3.3 till 3.7.
- **Section 3.3 Agility: the precursor** is added to investigate the roots of enterprise agility. This aids us in understanding how enterprise agility is emerged, and how it has developed overtime into a self-containing concept.
- **Section 3.4 Enterprise agility: the successor** further elaborates on the aspects of enterprise agility. This section describes all our literature findings, and explains how we rationalize these findings. The gap of knowledge is discovered in this section, and further elaborates on in the following section.
- **Section 3.5 Technology agility** zooms in on the technological aspect of agility. In addition, this section enables us to establish a link between enterprise agility and virtualization technologies.
- **Section 3.6 Virtualization technologies** describes what this concept comprises of, and how it has evolved overtime into differentiating, modern solutions. As containers originate from, and are also a form of virtualization technologies, we consider it interesting to know the details of virtualization. Moreover, these details are required so that we are able to compare performance and functionality of containers with conventional virtualization technologies (VMs).
- **Section 3.7 Containerization** focuses on the domain of containerization. This section incorporates on the history of containers, its architecture, main advantages and disadvantages, and other related aspects. The elaboration of this section supports us in understanding the concept of containers.
- **Section 3.8 Differences between VMs and containers** is the section where we study scientific sources on the performance and functionality differences of both virtualization technologies. With this section, we intend to denote the overall improvements containers provide more than VMs.
- **Section 3.9 Software development lifecycle** is the first additional concept elaboration. As we chose SDLC as the use case for our artefact, we are required to understand this domain. Therefore, we included this concept into the literature review.
- **Section 3.10 Maturity (assessment) models** is also added to the literature review, as we determined to design a maturity model for containers. Therefore, we need to discover characteristics of maturity models and best practices.
- **Section 3.11 Microservices architecture** is studied, since we found that this architectural pattern possesses the characteristics that containers require to utilize their benefits. Hence, we want to know what this architectural pattern comprises of, its philosophy, and other related concepts that can be used to foster container usage in organizations.
- **Section 3.12 Enterprise architecture and Microservices architecture** answers how containers can be integrated into the EA of an organization.

3.2. Preface of literature review

"*The problem of how organizations can successfully deal with dynamic, unpredictable environments has been a topic of interest in both practice and academe for several decades*" (Roberts and Grover, 2012). Enterprises weaponized themselves against this kind of environments by becoming agile. Agile enterprises make use of what has been defined by the scientific and business domains as Enterprise Agility (EAG). EAG is a relatively new concept that is getting more attention in different fields. This is due to the increasing pace of technology, whereby markets becoming volatile, and customers expecting innovations from the market. Hence, the need for more agile responses has grown (Christopher and Towill, 2001). In 2006, EAG was defined for the first time in an article concerning EAG and the enabling role of IT (Overby et al., 2006). After that, EAG became gradually a self-contained concept. However, EAG has its origins in another concept: Agile or Agility. This stepwise process of growth towards EAG is described in both section 3.3 and 3.4.

Section 3.5 includes an explanation about the link between EAG and technology as found during the literature review. This link is further specified towards describing virtualization technologies, as containers arise from this technology. Therefore, section 2.6 describes the history and state of the art of virtualization technologies, followed by an explanation about how containers emerged. After that, the concept of containers and their current use are described in section 2.7.

Figure 13 depicts the structure of the concepts and corresponding sections. Relations between each set of concepts, as shown in this structure, is explained towards the end of each section. However, each section can be individually read, as no prior knowledge of a former section is required to read the subsequent section.

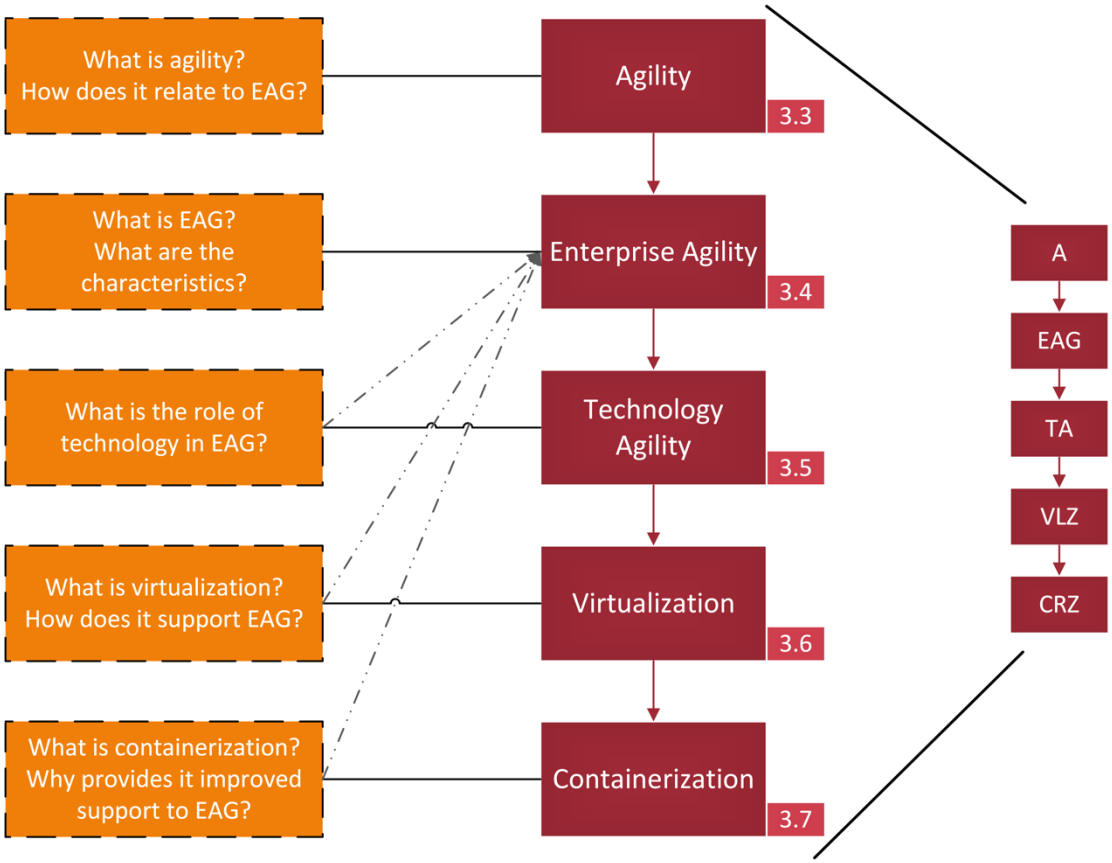
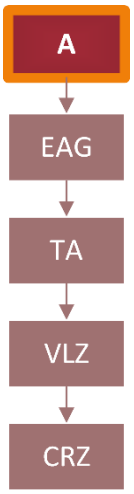


Figure 13: Literature review roadmap

On page 61, the second phase of the literature review starts. This is additional research towards the topics of SDLC, maturity models, and EA, which we considered necessary in order to answer the main RQ.

3.3. Agility: the precursor



Agility is a concept that was introduced by Iaccoca Institute of Lehigh University in 1991 after several workshops led by Nagel and Dove. At the time, the manufacturing domain in the US experienced heavy competition from Western Europa and Japan, and therefore lost their leading position. In order for the US to regain competitiveness and their leading position, the Iaccoca Institute conducted research in the field of manufacturing, and found that agility was key in achieving US' ambition (Nagel & Dove, 1991). The corresponding report contained descriptions about agile manufacturing enterprise, components, infrastructure, and operating mechanisms. Besides that, competitive foundation, characteristics, elements, and enabling subsystems of agility were additionally provided. This was the first scientific article where agility was mentioned and the beginning of a new research line/topic, besides the already existing LEAN manufacturing research line. Iaccoca's (Nagel & Dove, 1991) definition is provided in next box to give an understanding of the foundations of agility.

A system that shifts quickly among product models/lines, ideally in real time in order to respond to customer needs.

Citation/definition 3: Agility (Nagel & Dove, 1991)

Subsequently, more studies followed that explored agility. Most of them focus(ed) on the manufacturing domain, resulting in the Agile Manufacturing (AM) discipline. This discipline has been defined with respect to the agile enterprise, products, workforce, capabilities and the environment that gives impetus to the development of agile paradigm. Moreover, as a mark of the newness of the concept, every publication in this decade attempted to define and explain agility (Yusuf et al., 1999), resulting in a collection of definitions. These definitions share resemblance with one another, however, also differentiate when compared on certain levels.

Since all those definitions share similarities with (the development of) current *characteristics* of agility and EAG, some general findings from various authors are given (summarized by Yusuf et al. (1999)):

- High quality and highly customized products (Goldman & Nagel, 1993; Kidd, 1994; Booth, 1995; Hilton, Gill, Little, 1994);
- Products and services with high information and value-adding content (Goldman & Nagel, 1993; Goldman, Nagel, Preiss, 1995);
- Mobilization of core competencies (Goldman & Nagel, 1993; Kidd, 1994);
- Responsiveness to social and environmental issues (Goldman & Nagel, 1993; Goldman et al., 1995; Kidd, 1994);
- Synthesis of diverse technologies (Burgess, 1994; Kidd, 1994);
- Response to change and uncertainty (Goldman & Nagel, 1993; Goldman et al., 1995);
- Intra-enterprise and inter-enterprise integration (Vastag, Kasarda, Boone, 1994; Kidd, 1994; Youssef, 1992; Yusuf, 1996).

Considering these findings, main distinguished characteristics are:

1) High-end products and services;

2) Responsiveness to dynamic environment;

3) Optimal utilization of enterprise's assets, especially technology.

In addition, Ganguly, Nilchiani, and Farr (2009) created a table including eleven agility definitions along with the so-called essential characteristics embedded within those definitions. These characteristics are 1) Speed/time, 2) Cost, 3) Responsiveness, 4) Flexibility, 5) Quality, and 6) Customer needs, and are based on all findings of former studies concerning the attempts to define agility and its characteristics. When looking at the score, the definition of Yusuf et al. (1999) and Dove (1999, 2001) – 7 & 9 in Appendix I– have a positive score on all of the characteristics, resulting in the most complemented definitions. Therefore, the definition of Yusuf et al. (1999) is selected to further elaborate on. Besides that, in contrast with the first definition on agility by Nagel and Dove

(1991), the definition of Mathiyakalan et al. (2005) is also selected to elaborate on, since it is one of the newer definitions and additionally contains differences when compared. The literature review denotes a distinction between two decades: the first decade (1991-1999), and the second decade (2001-now). This distinction is used throughout the literature review.

3.3.1. Agility according to the first decade

Besides studying agility definitions, Yusuf et al. (1999) also embraced the denoted trend of attempting to define agility, which resulted in the following definition:

> *Agility is the successful exploration of competitive bases (speed, flexibility, innovation proactivity, quality, and profitability) through the integration of reconfigurable resources and best practices in a knowledge-rich environment to provide customer-driven products and services in a fast changing market environment.*

Citation/definition 4: Agility (Yusuf et al., 1999)

This definition denotes competitive bases, which have been identified in former studies, several characteristics that an agile organization should possess in order to realize outcomes on a certain quality standard, and the environment the organization is operating in. When compared to the definition of Nagel and Dove (1991), the following similarities and differences are found.

Table 3: Comparison of agility definitions of Nagel and Dove (1991) and Yusuf et al. (1999)

Similarities	Differences
An entity that acts with speed regarding their products.	Speed is solely focused on product lines/models, and not specified for the different aspects that are related to AM, as Yusuf et al. (1999) argues. Also, services are not mentioned.
Implying the importance of customer needs.	Iaccoca mentions that organizations should respond to customer needs. Yusuf elaborates on this by further specifying products into 'customer-driven' products.
	Yusuf mentions the competitive bases, based on former studies.
	The definition of Yusuf further denotes characteristics of an agile manufacturing enterprise, which are: integration of reconfigurable resources and best practices, enabling a knowledge-rich environment.

One of the differences is the level of detail of all mentioned aspects in regards to agility. Iacocca states the primal aspects of agility, whereas Yusuf provides a more detailed description of these aspects. This fact shows the gap of knowledge regarding agility around 1991, and denotes the contribution of all studies that have been conducted in the following eight years.

Yusuf et al. (1999) uses this scientific knowledge contribution of eight years in their paper. When further investigated, this definition shows several aspects (Yusuf et al., 1999):

1. It describes **agility as a system** (input, operationalization, and output), enabling a systematic approach for practitioners. Whereas organizations that apply agility have the possibility of using any existing methods and technologies for flexible manufacturing (Goldman et al., 1993). Besides that, Goldman et al. (1995) defined several outcomes for agility, examples are dynamic, context specific, aggressively change embracing and growth oriented, and winning market share and customers. Gehani (1995) and Kidd (1996) strengthen the outcomes, as they argue that agility is the ability to grow in a competitive market of continuous change, to respond quickly to rapidly changing markets driven by customer-based valuing of products and services.

2. Specification, exploration and subsequently **usage of competitive bases**. Ren, Yusuf, and Burns (2000) explored how agility attributes influence **competitive bases** of the enterprise. The authors defined these competitive bases as *dimensions that a firm's production system must possess in order to meet demands of the target market.* In this MSc thesis, these are defined as *Agility Characteristics* (AC).

3. The definition implies **three levels of agility**, which are agility for the individual, enterprise (Goldman et al., 1995), and inter-enterprise (Figure 11). Respectively, this means Elemental, Micro-agility, and Macro-agility (see Figure 3). According to Yusuf et al. (1999), the figure suggest that a truly agile organization focuses on individual *resources* (people, machinery, and management), and the *functions* that make up the enterprise to achieve the best possible output. It is also stated that it is the harmonization of these aspects of the organization that leads to agility (depicted by the orange arrow), rather than the individual optimization.

4. Finally, the definition denotes the **four main concepts of agility**, since these are increasingly being recognized as four key concepts for agile competition. These concepts include competition based on 1) Core competence management, 2) Virtual enterprise formation ('Inter-enterprise' in Figure 14), 3) Capability for reconfiguration, and 4) Knowledge-driven enterprise.

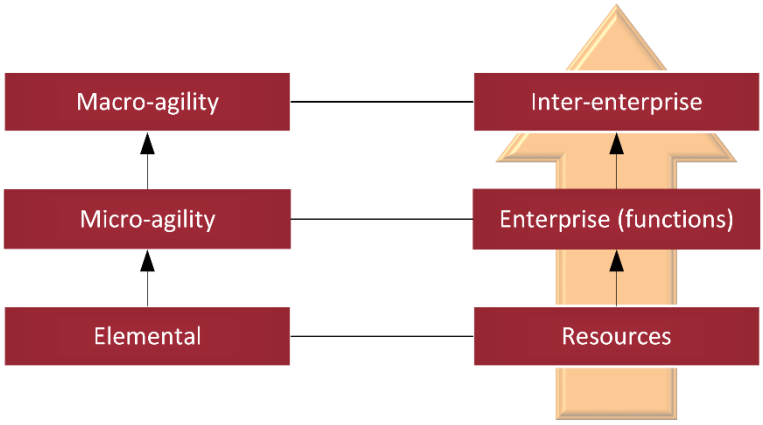


Figure 14: Hierarchy of agility of Yusuf et al. (1999) (edited)

In short, 1) Core competence management consists of two levels, individual and firm. On individual level, peoples' skill, knowledge, attitude, and expertise are included (Kidd, 1994). Regarding the level of the firm, core competences should bring strategic advantages and long-term benefits (Prahalad & Hamel, 1999).

2) A virtual enterprise has collaboration between different organizations on both management and operational level. This means that agile teams are working together with teams of external organizations, which realizes a "supply chain" of multiple core competences that are used to create products or services.

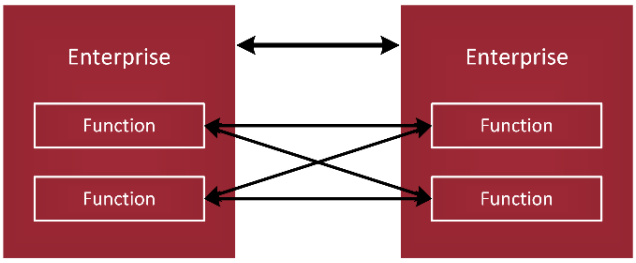


Figure 15: Virtual enterprises partnership development model of Yusuf et al., (1999) (edited)

Thirdly, agile enterprises should possess 3) the Capability to reconfigure themselves properly in a timely manner, by developing a strategic architecture featuring a corporate-wide map of skills (Prahalad & Hamel, 1999).

Finally, an enterprise should be 4) Knowledge-driven. Organizations that intend to become agile should include the development of a well-trained and motivated workforce, which possess the right set of skills, expertise and knowledge, as an essential element of their strategies.

Most literature regarding the beginning of agility has been conducted during the 90s. Yusuf et al. (1999) summarized all that literature at the end of this decade, which gave the authors an extensive amount of available information to process. This resulted in an overview that consists of ten decision domains of an agile manufacturing enterprise, and 32 related agility attributes. Table 4 denotes these attributes. In section 2.3.2, this table is used to link the definition of EAG.

Table 4: Decision domains and related agility attributes

Decision domain	Related agility attributes
Integration	Concurrent execution of activities; Enterprise integration; Information accessible to employees.
Competence	Multi-venturing capabilities; Developed business practice difficult to copy.
Team building	Empowered individuals working in teams; Cross-functional teams, Team across company borders; Decentralized decision making.
Technology	Technology awareness; Leadership in the use of current technology; Skill and knowledge enhancing technologies.
Quality	Quality over product life; Products with substantial value addition; First time right design; Short development cycle time.
Change	Continuous improvement; Culture of change.
Partnership	Strategic relationship with customers; Close relationship with suppliers.
Market	New product introduction; Customer driven innovations; Customer satisfaction; Response to market changes.
Education	Learning organization; Multi-skilled and flexible people; Workforce skill upgrade; Continuous training and development.
Welfare	Employee satisfaction.

The study of Yusuf et al. (1999) is selected as one of the main sources for agility of the first decade to use in this document, as it investigates multiple sources of agility's first decade (1991-1999) to support their own findings and increase the validity of their results. These used sources have been studied for the Long Proposal to validate their correctness and to be able to link the sources to the statements.

However, in the same year, Sharifi and Zhang (1999) conducted a study on agility as well. This is study is important due to several reasons. First, these authors defined agility drivers, providers, and capabilities, based on prior literature. The agility capabilities are (Sharifi & Zhang, 1999):

1) **Responsiveness**: the ability to identify changes and respond fast to them, reactively and proactively, and recover from them;
2) **Competency**: the extensive set of abilities that provide productivity, efficiency, and effectiveness of activities towards the aims and goals of the organization;
3) **Flexibility**: ability to process different products and achieve different objectives with the same facilities;
4) **Speed**: the ability to carry out tasks and operations in the shortest possible time.

These capabilities form a foundation that was generally accepted as the main characteristics of agility, and are consequently used throughout all succeeding studies.

Secondly, Sharifi and Zhang (1999) were the first to mutate the demeanor of agility in their definition. Therefore, being the first scientists that argued that organization's that apply agility should be proactive in detecting changes, instead of reactively responding quickly to changes when they already occur. This is noticeable in the description of the capability 'responsiveness', and also in the author's provided definition:

> *Agility is the ability to **detect** the changes in the business environment, and respond to them by providing the appropriate capabilities.*

Citation/definition 5: Agility (Sharifi and Zhang, 1999)

Furthermore, 1999 was an important year for the field of agility, as Dove also complemented the body of knowledge regarding agility and its demeanor. Dove (1999) too denoted the difference between reactive and proactive modes of organizations, and subsequently recognized change proficiency in both. The author defined

reactive as an opportunistic change, and responds to a situation that threatens viability. Whereas *proactive* mode is defined as an innovative change, and responds to a possibility of leadership (Dove, 1999). Next sub-section further elaborates on this topic.

3.3.2. Agility according to the second decade

At the beginning of the new decade, Dove (2001) wrote a book about Response Ability. Here, the author focused on describing how the culture, structure, and language of the agile enterprise works. In this book, Dove continued to explain the idea behind the proactive demeanor. Figure 16 briefly depicts the differences in change proficiency.

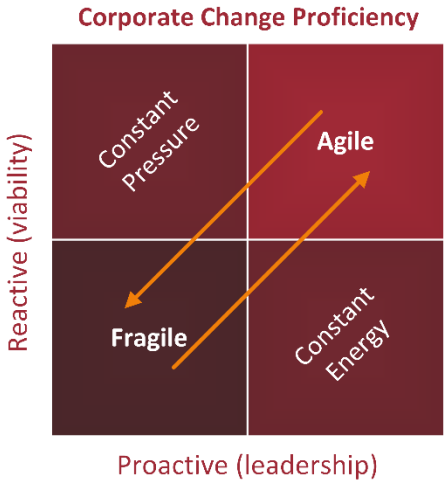


Figure 16: Change proficiency - Dove (2001) (modified)

However, the book did not provide an explicit meaning on how the proactive demeanor should be conceptualized and applied.

Another definition is provided to strengthen the current description about agility. This definition origins from 2005, the second decade, and therefore complements the overall perspective on agility in this literature review. The definition is created by Mathiylakan et al. (2005) and contains several similarities in comparison with definitions of the first decade. Although, this definition anticipates on the change proficiency differences and therefore shows an important difference.

> *Ability of an organization to detect changes (which can be opportunities or threats or a combination of both) in its business environment and hence providing focused and rapid responses to its customers and stakeholders by reconfiguring its resources, processes and strategies.*

Citation/definition 6: Agility (Mathiyakalan et al., 2005)

All definitions before 2005 (except Sharifi & Zhang (1999)) mentioned that one of the main aspects of agility is an organization solely *responding* to its dynamic environment. In Mathiyakalan et al.'s (2005) definition, one of the first notable differences is that the authors complemented this definition with 'the ability of an organization to **detect** changes' in its dynamic environment. This addition changes the demeanor of agility to proactive, instead of the former reactive demeanor. In the same year, Gartner Research Group (Ashrafi et al., 2005) also defined agility. Moreover, these researchers also complemented agility with "*An organization's ability to **sense** environmental changes…*". Both of these definitions are in line with Dove's (2001) earlier definition of agility (see Appendix I), where the author prepares the demeanor to be transformed, by defining agility as the ability of organizations to respond effectively and efficiently, although now "*to both **proactive** and reactive needs and opportunities on the face of an unpredictable and uncertain environment*". Hence, the proactive demeanor conceptualization to sensing and detecting was embraced and supported by a broad amount of scientists. To support the importance of this shift of demeanor, Ren, Yusuf, and Burns (2003) revealed that the competitive bases speed, flexibility, and **proactivity** had the largest impact on the overall enterprise's competitiveness.

3.3.3. The overall view – Growth of definitions

The aforementioned table of Ganguly et al. (2009), containing known definitions of agility, is complemented with additional definitions found during the literature review, and is provided in Appendix I – Agility definitions. Some definitions and corresponding authors and years were found to be incorrect, and are therefore manually modified. During the analysis of all the definitions, several notable findings have occurred.

The table contains 16 agility definitions, ranging from 1991 to 2005. In 1991, agility was defined for the first time. It was argued that a system should be able to shift quickly between product lines, and to subsequently respond to customer needs, stating the importance of the customer's role. After that, Kidd (1994) continued with the characteristic of speed by mentioning a "*rapid and proactive adaption of enterprise elements to unexpected and unpredicted changes*", embracing the prior definition and complementing it with the impacting

role of an organization's environment. One year in advance, the characteristic competitiveness made its entrance into agility, combined with the recognition of the continuously changing customer habits (Goldman et al., 1995). The definitions that followed after 1995 further specified the (more) abstract characteristics of the prior definitions, resulting in more comprehensive definitions. During 1991-1995, the average word count per definition was 15, whereas the average word count between 1996-1999 increased to 29. The increasement contained further specifications about the competitive environment, trend of changing markets, and the more influencing role of customers. Moreover, Fliedner and Vokurka (1997) defined agility as the ability to market products successfully, but accompanied with the characteristics of low-cost, high quality, short lead times, and varying in volumes that realize enhanced value to customers through customization. Meaning that organizations should focus on becoming (more) customer-driven in order to increase value to its customers. Eventually, Yusuf et al. (1999) used the competitive bases in their definition (section 2.2.1), acknowledging these bases as definitive characteristics of agility. In the same year, Sharifi and Zhang (1999) defined the first agility attributes, and were the first to suggest changing agility's demeanor in their definition.

Figure 17: Agility timeline: abstract representation

During the transfer of the millennium, Dove (1999, 2001) published two scientific contributions where the author argued that organizations should respond efficiently and effectively to both reactive and proactive needs and opportunities of the unpredictable environment. This was the embracement of the proactive demeanor of agility, and the second step of transforming agility from solely reactive responses, to proactively anticipating on upcoming changes (grey area in Figure 17). After that, Sambamurthy, Bharadwaj, and Grover (2003) denominated the market environment as highly dynamic, besides being unpredictable. Implying that the market and customer needs are getting more dynamic than before. Subsequently, the Gartner Research Group (McCoy & Plummer, 2006) and Mathiyakalan et al. (2005) conceptualized the embraced proactive demeanor of Dove into the organizational ability of 'sensing' and 'detecting' upcoming environmental changes as a new characteristic of agility, continuing with the idea of Sharifi and Zhang (1999).

3.3.4. The overall view – Differences and similarities

Sub section 2.2.3 provides an explanation of the known agility definitions. The definitions show an incremental growth of the concept of agility. When studied, a pattern is revealed where agility characteristics are at first defined in an abstract manner (e.g. 'changes' and 'proactive') and several years later conceptualized to tangible concepts (e.g. respectively 'unpredictable environment' and 'sensing environmental changes').

Apart from this incremental process, the definitions do contain some differences when compared. As most definitions define agility as an 'ability' (9), some definitions define agility as a 'capacity' (1), as a 'capability' (2), or use a more abstract concept (an adaption, the exploration of, and a system). Deep analysis of these concepts is out of scope. However, brief analysis of capacity's and capability's definitions show a matching description of the concept as "the *ability* or power to do something". Therefore, defining ability as the parent/higher concept of them.

As aforementioned, all definitions differentiate from each other on a higher detail level. However, despite the differences, all definitions of 'agility' emphasize the speed and flexibility as the primary attributes of an agile organization (Gunasekaran, 1999; Sharifi & Zhang, 1999; Yusuf et al., 1999). An equally important attribute of agility is the effective response to change and uncertainty (Goldman et al., 1995; Kidd, 1994; Sharifi & Zhang, 2001). Sharifi and Zhang (1999) emphasize this, as they state that responding to change in proper ways, and exploiting and taking advantages of changes are the main factors of agility. Additionally, Sherehiy et al. (2007) states that the next common component of published definitions of agility is a high quality and highly customized products (Gunasekaran, 1999; Kidd, 1994; Mccarty, 1993; Tsourveloudis & Valavanis, 2002). Figure 18 denotes the selected agility characteristics according to this project's conducted study.

Figure 18: Agility characteristics - Based on Sharifi and Zhang (1999) (complemented)

3.3.5. Preliminary conclusion of agility compared to enterprise agility

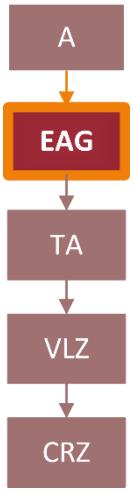
Whereas agility has its focus on responding properly to change, EAG[4] incorporates the sensing of possible upcoming changes and then developing a proper response, as is argued by McCoy and Plummer (2006) and Mathiyakalan et al. (2005). Moreover, as mentioned before, 11 out of 16 definitions think of agility as an ability of an organization. However, EAG is seen as a way of working (WoW) that is applied throughout the whole enterprise, instead of being applied in certain parts of the organization.

Moreover, the main difference between both concepts is that most studies on agility are focused on, and therefore specialized for, the field of manufacturing. EAG goes further and expands the agile WoW throughout the whole enterprise, instead of 'detaining' it to one function or specific industry. This is, however, performed by making use of the aspects that are formerly found by and applied at all prior manufacturing-oriented studies. These studies defined agile attributes and aspects, and showed a relationship between development to the current state of EAG. Section 3.4 elaborates on the development of EAG and depicts both the similarities and additions of agility in regards to EAG.

The above-explained concepts about agility are described in the Long Proposal, since these specific components share resemblances with characteristics of enterprise agility. Agility is the precursor of EAG, which means that the development process of agility is essential to know, in order to understand the specific growth of EAG. Concluding, it is stated that agility goes beyond solely speed (Youssef, 1994), response, and flexibility (Kidd, 1994), and is the full use of the developed and well-known technologies, combined with all the lessons learned from existing lean methods and production management philosophies (Goldman & Nagel, 1993; Yusuf et al., 1999).

[4] As aforementioned in section 3.1, EAG is the abbreviation of 'Enterprise Agility'.

3.4. Enterprise agility: the successor



During the second decade, studies on agility that focused on AM, were increasingly extending their scope regarding the enterprise-wide application of agility. This resulted in different sub-types of agility. Moreover, this is seen as the further developed form of agility, what eventually led to the all-encompassing concept of EAG. Most significant sub-types are Organizational agility and Workforce agility. Other less widely used sub-types are business agility (Mathiyalakan et al. (2005), customer agility, partnering agility, and operational agility (Sambamurthy et al., 2003).

Sherehiy et al. (2007) conducted a review on enterprise agility and its origins. Based on all studied literature, the authors state that in reference to agility, the most important enterprise components are Organization, People, and Technology[5]. Each of these elements is multidimensional and complex itself (Sherehiy et al., 2007). This means that EAG should incorporate these three enterprise components in its application throughout the organization. Therefore, in order to state a definition for EAG in this MSc thesis, specific agility descriptions aimed at OPT are used.

Returning to the emerged sub-types of agility. O and P have linked sub-types of agility, as can be seen in Table 5. Sub-types denoted as significant (**bold**) represent the enterprise component as stated by Sherehiy et al. (2007), which have dedicated research to that agility sub-type. However, when aligned after each other, it shows a gap of knowledge regarding the lack of conducted studies on the technological aspect of agility (*italic* and underlined).

Table 5: Enterprise Agility academic (red) and business (orange) definitions

Agility sub-type	Description	Enterp. comp. (OPT)	Examples of sources
Organizational agility	Agility aimed at the Organization	Fulfills **O**	Reed and Blunsdon (1998).
Workforce agility	Agility aimed at the People	Fulfills **P**	Forsythe (1997); Plonka (1997).
Agility for technology	Agility aimed at Technology	…	…

Findings of the literature review showed that there is no explicit link between agility and technology. Moreover, no research lines are dedicated to agile technologies, as is done with Organization and People. Figure 19 visualizes this finding. This missing link may indicate a gap of knowledge regarding agility and technology. Section 3.1.3 further elaborates on this finding. Next sub-sections describe O and P of the agility sub-types.

Figure 19: visualization of missing link to Technology in EAG

3.4.1. Organizational agility (Organization)

Organizational agility is described as *an organization's capacity to adjust its internal structures and processes in response to changes in the environment* (Reed & Blunsdon, 1998). Additionally, Nejatian and Zarei (2013) describe that organizational agility is closely related to the concepts adaptability and flexibility, and all three concepts are interchangeably used to indicate the endeavors made by an organization for handling dynamic markets and unpredictable changes. Moreover, internal adaptability and flexibility are two main characteristics, what are vital for the evolution of organizations towards achieving organizational agility (Nejatian & Zarei, 2013). Sherehiy et al. (2007) argues in their review that the highest level of development is reflected in form of organizational agility, which comprises both concepts of adaptability and flexibility. IT also contributes in organizational agility, as it increased organization's capacity to manage change, and led to greater investments in assets supporting the efficiency of specific organizational structures (Boer & Van Engers, 2013). Finally, the ultimate goal of organizational agility is to

[5] From this point, the combination of Organization, People, and Technology will be referred to as "OPT".

achieve a higher market share in a competitive environment by modifying internal structures to anticipate on environmental changes.

3.4.2. Workforce agility (People)

Workforce agility is focused on the employee workforce of an organization. The main outline of agile workforces is aimed at possessing and leveraging highly trained and skilled employees (Youndt, Snell, & Dean, 1996), what leads to the benefits of quality improvement, better customer service, learning curve acceleration, economy of scope and depth (Herzenberg, Alic & Wial, 2000; Hopp & Van Oyen, 2004). Gunasekaran (1999) proposed the following requirements to improve organization's workforce: 1) closer interdependence among activities, 2) different skill requirements, 3) more immediate and costly consequences of any malfunction, 4) output more sensitive to variations in human skill, knowledge and attitudes, and to mental effort rather than physical effort, 5) continual change and development, 6) higher capital investment per employee, and 7) favor employees responsible for a particular product, part, or process (Pinochet, Matsubara & Nagamachi, 1996). In addition, the agile workforce is expected to provide fast response to unexpected events (Plonka, 1997), and also to effectively take part in any collaborative environment (Forsythe, 1997). This workforce can be located throughout cross-functional project team, collaborative ventures with external parties, or a virtual organization (enterprise) (Van Oye, Gel & Hopp, 2001).

Several authors also define specific workforce agility attributes, which share similarities with agility attributes. Breu, Hemingway, and Strathern (2002) used two agility dimensions, speed and flexibility, to assess these attributes. The study showed that most important attributes for workforce agility are 1) speed of developing new skills, 2) responsiveness to changes in customer need and market environment, and 3) speed of acquiring the skills needed for business process change. Eventually, the attributes were categorized into five higher levels (after Breu et al. 2002). In short:

- *Responsiveness*: in regards to changing customer needs and market conditions;
- *Competencies*: speed of acquiring, developing, and innovating several kinds of new skills;
- *Collaboration*: effectiveness of cooperating across functional boundaries and ease of moving between projects;
- *Culture*: employee empowerment for independent decision making;
- *Information Systems (IS)*: support of IT infrastructure for the rapid introduction of new IS.

Demeanor is also essential in Workforce agility. The aforementioned studies already implied proactive behavior applied by employees, and in 2003, Dyer and Shafer (2003) embraced this behavior by specifying proactive behavior of employees as workforce agility attributes.

These categories of agility workforce attributes are perceived in this MSc thesis as a specific part of EAG, as it describes the people aspect of EAG, based on OPT.

3.4.3. Inappropriate term usage

The appearance of these new agility fields also resulted in inconsistent use of the definition of agility. It occurs that 'enterprise agility' is mentioned where agility is meant, or the other way around. The same applies for the other sub-types of agility. One of the reasons is that most sub-types do not have widely accepted definitions, as agility does. Hence, resulting in authors that refer to agility sub-type X, but use the conventional definition of agility throughout the study, or the other way around. Especially in the beginning of the second decade, there was no clear distinction between agility and EAG. Moreover, despite the fact that EAG discussions and definitions were first seen in the second decade, EAG was already mentioned during the first decade. Although the meaning was rather different. Appendix II – Retrospective view on Enterprise Agility explains the EAG term usage in the first decade. One more recent example is given in this section, since it also shows relevance with current EAG. After that, the conclusion of this analysis is given.

Tsourveloudis and Valavanis (2002) developed a measurement of agility. The authors combined four areas (Production, Market, People, and Information), what are named as infrastructures, as the assessment of an organization's agility. However, Tsourverloudis and Valavanis (2002) specified the combination of these four

areas as **enterprise agility**, but whereas the assessment is measuring 'agility'. This is one of the situation where both concepts are used inconsistently.
Although, the authors state that the assessment evaluates the *overall agility of the enterprise*, implying a vision on the extension of agility throughout the whole organization, instead of agility detained to specific enterprise components. Nevertheless, when considering the four areas, the areas contain similarities with Sherehiy et al. (2007) statement about essential enterprise components (OPT). Market is added, which likely denotes the sensing of upcoming changes in the dynamic market. This is the first time that researchers stated 'enterprise agility' as an outcome in research towards agility.

The results of this brief analysis on EAG term usage (Appendix II) show the first use of the term 'enterprise agility' in literature. Most of the times, agility of an enterprise is meant accompanied with corresponding reactive demeanor, agility applied in a specific enterprise function, and main focus on the field of manufacturing. 'Enterprise agility' in terms of EAG, accompanied with corresponding proactive demeanor, agility applied enterprise-wide, and the ability of sensing, is not meant. However, Dove (2001) does refer to EAG in its description of the agile enterprise. Although this is not explicit, it does correspond with found EAG components.

3.4.4. Enterprise agility according to academic and business literature

Frequently, the concepts agility, EAG, and agility sub-types are used incorrectly when referred to. This is due to lack of widely accepted definitions in the scientific domain. Therefore, solely the definitions that refer explicitly to "*enterprise agility*" are incorporated into the definition table of EAG. Definitions that refer to "agility" are excluded from the table. As only two definitions are found in the scientific field, the table is complemented with definitions from the business literature. This field clearly distinguishes enterprise agility from agility, and shows an expertise with the field due to the current popularity. The following table provides the found EAG definitions.

Table 6: Enterprise Agility academic (red) and business (orange) definitions

#	Author(s) and year	Definition
1	Overby et al. (2006)	The ability of firms to sense environmental change and respond readily.
2	Gartner Research Group (McCoy and Plummer, 2006)	The ability of an organization to sense environmental change and respond efficiently and effectively to that change.
3	Sogeti (Van Herpen, 2014) *Paraphrased*	Combined application of strategy alignment, organizational and culture fit, collaboration through the lifecycle of products and services, and integration with external parties, so that agility is present throughout the entire value stream.
4	PwC US (PricewaterhouseCoopers, 2015) *paraphrased*	The successful combination of the operating environment (maintaining of USP while critically scanning for risks and opportunities), strategic responsiveness (to develop an appropriate response quickly), and organization flexibility (internal alignment of the new strategy for execution), in order to sense change and respond rapidly.
5	Korn Ferry (2014)	Enterprise agility is a company's ability to outperform the competition and drive growth in new, ambiguous situations by learning and adapting when confronted with foreseen and unforeseen circumstances, dilemmas, crises, and complex problems.
6	Deloitte (Oliver and Muir, 2017)	An organization's ability to respond to, adapt to, and even lead the market when it comes to customer centricity, employee experience, external market and competitor movements, industry or regulatory changes, internal changes, and technology innovation and advancement.
7	Modern Analyst (Alami, 2017)	The ability to adapt easily to change. It is the ability and capability of a system to respond rapidly to a certain modification by adapting its inceptive and stable configuration.
8	(McKinsey&Company, n.d.) *Paraphrased*	A combination of stability, which provides direction and an efficient and scalable backbone, and dynamism, what comprises a network of dynamic teams that radically increase speed, flexibility, and ownership.

A difference between business and academic definitions is the classification of enterprise agility. Similar to all agility definitions, academic EAG definitions state that EAG is an *ability* of an organization. Three out of six business definitions, however, do not mention any classification. This indicates that there is currently no commonly accepted classification of EAG.
Besides that, business definitions show the same elements as the younger academic agility definitions in Appendix I. This strengthens the prior statement about the non-existence of a clear transition between agility and EAG, and subsequently indicates the conclusion of an incremental process.

Boer and Van Engers (2013) provided another definition of agility, which is interesting in regards to EAG.

> *Agility is not an infrastructural concept. It is a matter of awareness of the organizational environment and the ability to foresee problems and react appropriately swiftly.*

Citation/definition 7: Agility (Boer & Van Engers, 2013)

This definition is intentionally described in current subsection about EAG definitions, since it mentions a specific characteristic that other authors did not provide: "*…matter of **awareness** of the organizational environment…*". The authors build forth on the demeanor of proactive and reactive, stating that organizations should additionally be aware of their environment.

3.4.5. Transition to Enterprise Agility

Despite all the written academic and business literature on agility related topics, Sherehiy et al. (2007) states that there is no commonly accepted definition of EAG, and that there are a large number of opinions concerning the meaning of this term.

However, according to Overby et al. (2006), EAG can be broken into two (enabling) components: sensing and responding. This is in line with the description of agility in section 3.3.1. It also implies the incremental difference between agility and EAG that has been developing since the addition of detecting upcoming environmental changes.

Besides that, Sherehiy et al. (2007) defined main agility attributes (besides the *core* agility attributes as in section 1.1.4), which are: Flexibility, Responsiveness, Speed, Culture of change, Integration and Low complexity, High quality and customized products, and Mobilization of core competencies. These seven attributes share some resemblances with the stated agility attributes in Figure 18 (p37). Although, a noticeable difference is the addition of culture of change, which implies the relationship with workforce agility (People). The authors additionally mentioned that the main attributes can be applied enterprise-wide, referring to the application of agility throughout OPT.

As overall analysis and the table of both agility and EAG definitions show, there is no moment of change or transition from agility to EAG. Therefore, the switch from agility to EAG is an incremental process that is executed throughout several years[6]. Based on studied academic and business literature, and in contrast to agility, EAG differentiates itself from agility through the following components:

- The ability of sensing upcoming environmental changes (Proactive demeanor);
- Agile WoW on an enterprise-wide scale, meaning agility attributes applied into all enterprise components, resulting in all departments (e.g. Sales, HR, Finance) applying the agile WoW;
- Agility applied in the multidimensional levels of OPT.

The underlying Figure 20 visualizes the difference between agility and EAG.

[6] The process of Agility to EAG is incremental. However, this explanation does not aim to describe that agility is going to be completely replaced with EAG. The self-contained concept of agility stands on itself.

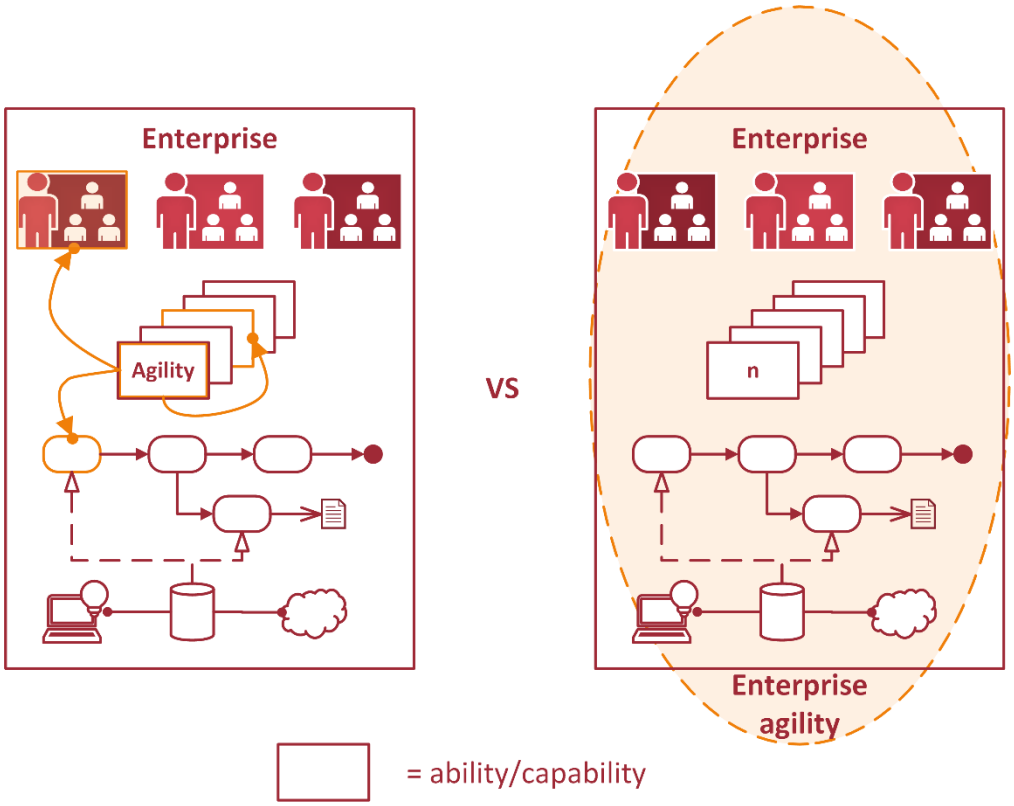


Figure 20: Visualization of Agility vs EAG

Therefore, considering Overby et al. (2006) split into two enabling components, Sherehiy et al. (2007) main agility attributes, and these differentiators, the following stipulative definition of EAG is created, in combination with all aforementioned academic and business literature.

> *The collection of principles that enable an enterprise to sense environmental changes in a fast, unpredictable, dynamic business environment and to provide rapid and accurate responses effectively and efficiently, using all competences of the enterprise to internally align, organize and realize this change, while aiming to continuously improve its workforce, and focusing on optimal utilization and exploration of technology.*

Citation/definition 8: EAG (primarily based on Dove, 2001; Mathiyakalan et al., 2005; McCoy & Plummer, 2006; Sherehiy et al., 2007)

To clarify the structure of the definition, specific parts are abstracted to its type. *Italic* means that the concerning elements are in general represented throughout the definition. This resulted in the following:

> [Classification of EAG] – [*Demeanor*: Proactive] – [*Demeanor*: Reactive] – [Agility characteristics application: Enterprise-wide] ~ [*Interwoven OPT dimensions*].

Above definition is used for the remaining of this master thesis project. The overall timeline of Agility and EAG can be seen at page 43.

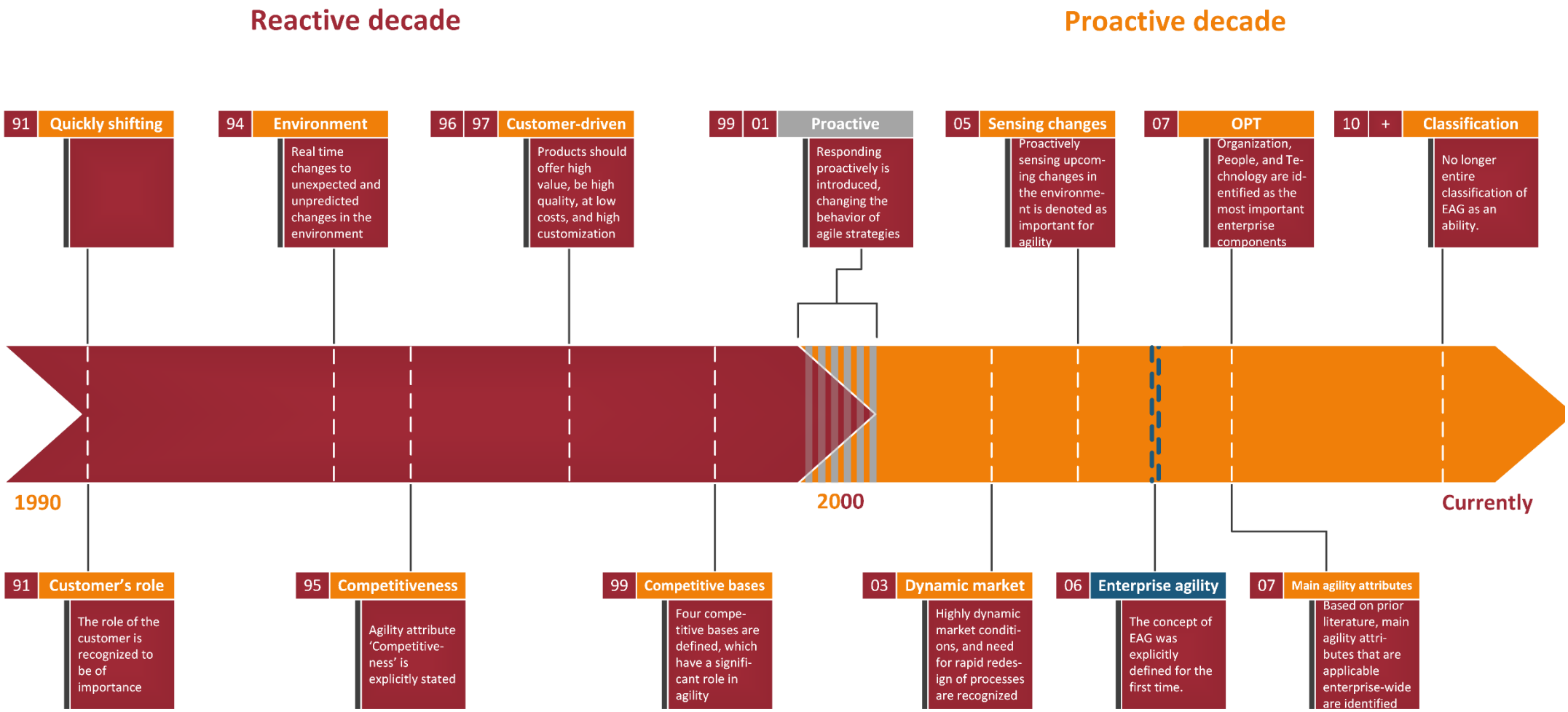
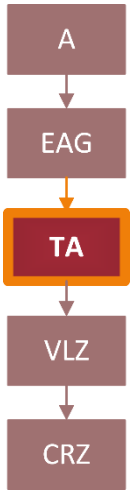


Figure 21: Agility timeline

3.5. Technology agility



This section elaborates on literature around technology and agility. In addition, the section focuses on explaining the literature review results of agility and EAG, and linking these to technology. Based on this explanation, the bridge between agility and technology is made.

3.5.1. Literature review findings

Based on the results of the literature review, no explicit link is found between EAG and technology. There are, however, associations noticed between agility and technology. During the execution of the literature review, it was found that most studies still use IT as a supportive element to all agility types in enterprises. Despite the wide recognition of the crucial role of IT nowadays, no study created a self-contained sub-type of agility with focus on IT or technology.

However, technology is several times associated with agility. For example, the IT landscape should be adaptive according to multiple authors (Pal & Pantaleo, 2005; Kaddoumi & Watfa, 2016). Besides that, Tseng and Lin (2011) have developed a conceptual model of enterprise agility that encompasses of agility drivers, capabilities, and providers. These authors argued that *Technological innovations* is one of the main agility drivers, and *information integration (infrastructure)* is an important agility provider. In addition, Yusuf et al. (1999) states that in reference to agility, an organization should be aware of arising technologies, pursue leadership in technology use, and obtain skill and knowledge regarding technology.

The technology domain also noticed an increasing demand of agile software solutions. These solutions were required to support an adaptable IT landscape and facilitate an agile WoW. Moreover, the literature review additionally showed that several authors argued that the following characteristics should be considered for the configuration of agile IT:

- **Technical integration** (Sharp, Irani, and Desai, 1999; Lee, Siau and Hong, 2003; Pal & Pantaleo, 2005; Holbeche, 2015; Rigby, Sutherland, and Takeuchi, 2016).
- **Standardization** (Lee et al., 2003; Pal & Pantaleo, 2005; Sherehiy et al., 2007).
- **Reusability** (Dove, 1999; Yusuf et al., 1999; Sherehiy et al., 2007).
- **Scalability** (Dove, 1999; Yusuf et al., 1999; Sherehiy et al., 2007).

Software product organizations noticed this increasing demand, which resulted in rapid improvements in the field of technology that enabled organizations to improve their agility. Nevertheless, none of the scientific sources made an explicit link between existing technologies and agility. This denotes that there is a gap of knowledge regarding an explicit link between EAG and technology.

3.5.2. Technology towards agility

As aforementioned in Section 3.4, agility research lines dedicated to the O & P enterprise components exist. Another well-known area where agility has shown its effectiveness is software development. However, when observing the domain of technology, again no examples are found in the literature where agility is explicitly applied in or linked to technology.

Although, when looking to the infrastructure domain, several technological innovations have occurred that made the infrastructure more agile. Best-known example is the introduction of virtualization. This technology enabled organizations to divide the computing resources of a single server, realizing a higher server utilization. Basically, virtualization made organizations use their infrastructure in an agile way.
Another innovation was the introduction of the cloud. This provided organizations identical computing resources through the internet, removing the need for organizations of having and maintaining their own server park. Different types of virtualization technologies enable the cloud concept, and the amount of organizations using a cloud solution have significantly increased since the cloud's introduction. The results of the utilization of virtualization show that this concept is the enabler of an agile infrastructure and technologies.

In 2013, another technological innovation has been introduced, which is containerization. Briefly stated, this technology provides the same services as virtualization. However, it is facilitated faster and at the costs of fewer resources through different functionality. Moreover, the characteristics of containerization align with the attributes of agility, even more than the characteristics of virtualization (as shown in section 1.1.4). Besides all stated technologies perceived as agile, containers further improve functionality of virtual machines. This is interesting to apply in organizations to advance enterprise agility.

Currently, literature around containerization and its role is found. However, there is no link made explicit between containers and EAG (Figure 22). Therefore, next section elaborates on virtualization technologies, explaining how it emerged, its role, to eventually move on to containerization. With describing containerization, the hypothesis about the missing link between containerization and EAG is tried to prove.

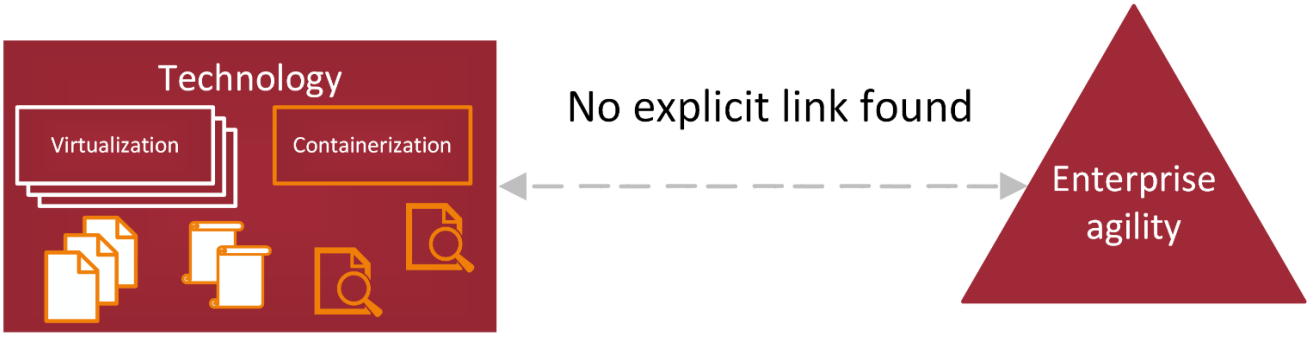
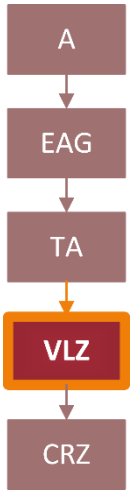


Figure 22: Visualization of missing link between EAG and containers

3.6. Virtualization



Virtualization is a technology that has become more widely used since its introduction. It is also the precursor of containerization, despite it being a different approach to software. This section describes the following aspects of virtualization and containerization:

- History of virtualization;
- State of the Art of virtualization;
- Virtualization and cloud computing;
- Virtualization advancement.

3.6.1. History of virtualization

Virtualization has its origins in the 1960s. Different technology organizations (e.g. General Electric (GE), Bell Labs, and IBM) were experimenting with sharing computer resources among large groups of users. In 1963, Massachusetts Institute of Technology (MIT) announced project MAC, which stood for Multiple Access Computer. In short, to be able to execute this project, MIT needed new computer hardware that was capable of supporting multiple users simultaneously, instead of the former single user support per computer (Conroy, 2018). Subsequently, MIT sent Requests for Proposals (RfP) throughout the technology domain. GE was willing to make a commitment to time-sharing computing (Conroy, 2018).

IBM noticed the increasing demand for such systems, and started its own projects regarding time-sharing computing development. Time-sharing computing means that multiple users are making use of the same computer by means of multiprogramming, realizing an efficient use of the computing power and resources.

The next years, all developments resulted in computers that enabled organizations to let multiple users work on a single computer. More specifically, IBM designed the first commercial mainframe (CP-67) that supported virtualization (Conroy, 2018). Virtualization was a more advanced type of software concept when compared to the former time-sharing concept. The idea was that specific software (Control Program (CP)) ran on the mainframe and created virtual machines (VM), which is the isolated environment where users would interact with as their personal computer. One of the biggest advantages was that virtualization enabled organizations to configure multiple OSs on single servers (Carroll, Kotzé & Van der Merwe, 2011). Other main advantages that virtualization provides in comparison to time-sharing computing are:

Table 7: Advantages of virtualization compared to time-sharing

Description
More efficient system usage. VMs share the overall resources of the mainframe, instead of equally divide resources between all users.
Improved security, as each user works in a complete isolated operating system (OS)
Improved reliability, as no user could crash the entire system, only their personal OS.

Next sub-section 2.2.2 elaborates on the state of the art of virtualization. This includes a selected definition that is further used throughout this research project.

3.6.2. Virtualization: State of the Art

After the introduction of CP-67, the technology has improved significantly and led to wider usage by organizations. More software emerged that provided support for multi-OSs, including the current widely used UNIX OS, and different types of virtualization were introduced. Around 2004, server virtualization was becoming mainstream and became increasingly more often implemented into organizations (Connor, 2004). Additionally, the author also stated that virtualization in general is becoming mainstream, especially since Microsoft entered the market, and because of the virtualization providers' race to develop solutions that effectively use the continuously increasing CPU power to a full extend (Connor, 2004). After that, Spiegel (2006) argued that businesses could benefit from application and server virtualization. The benefit is that applications could be installed on servers instead of being installed on local desktops, realizing secure and remote access. Due to

these concepts, organizations became aware of the potential savings on hardware purchase and maintenance costs. This eventually led to the transition to the cloud, which is further described in sub-section 2.6.3.

When referring to existing literature, there are large differences about the definition of virtualization (Luo, Yang & Ma, 2011). This is because most definitions are focused on specific virtualization types, instead of the general concept of virtualization. Luo et al. (2011) state that in the virtualization's development stage, no strict standards and definitions were created. Nonetheless, based on all business definitions and literature, the following stipulative definition has been composed as an attempt to provide a general definition.

> *Virtualization refers to technologies designed to provide a layer of abstraction between computer hardware systems and the software running on them, where system resources are logically divided in order to create a virtual resource.*

Citation/definition 9: Virtualization (based on Clark, 1965; Waters, 2007; Techopedia, n.d.)

"*Virtualization as an abstraction of hardware has become a popular solution to resource isolation and server consolidation*" (Li & Kanso, 2015). Due to this, many virtualization types and different virtualization providers[7] exist. Virtualization type examples are full virtualization, paravirtualization, and server virtualization. A comprehensive overview is included in Appendix III – Virtualization types.

Virtualization consists of three main characteristics: 1) Partitioning, 2) Isolation (Tamane, 2015), and 3) Server efficiency.

- **Partitioning**: one physical system can support multiple applications and OSs by partitioning available resources.
- **Isolation**: each created VM operates without affecting its host OS and other parallel VMs, as VM-specific data is not shared between one another. This additionally improves security.
- **Server efficiency**: virtualization creates virtual partitions of servers, realizing efficient use of servers.

3.6.2.1. Virtual machine architecture

The construction of a VM is showed in the right Figure 23. The basic construction of a VM environment is depicted to visualize the construction. In the next paragraph, the figure is explained per layer (bottom-up).

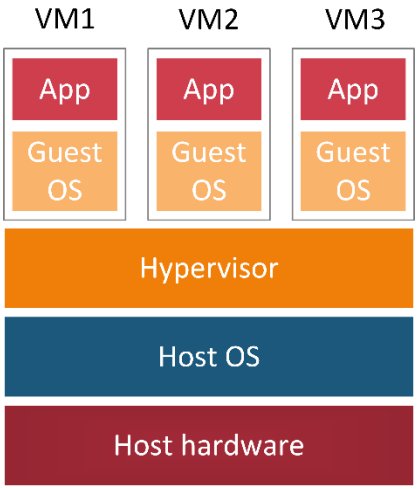


Figure 23: VM architecture

The lowest layer (dark red) consists of physical hardware. Onto this hardware is the host OS (blue), which is the main system software of that particular set of hardware. This OS manages all the tasks and activities that are requested from the hardware. The third layer is the hypervisor (orange). This is a software component that emulates the existing hardware of that environment as individual hardware components for the VMs. Due to this, a VM thinks that it is using its personal CPU, memory, storage etc., whereas it is actually using a part of an extensive hardware component or a large set of hardware. This enables given configuration of different VMs with the same (set of) hardware. Inside a VM, an OS (Guest OS) is executed, which results in the new environment where the desired application or service is ran in. Abstractly formulated, basic virtualization realizes a computer (Guest OS) inside another computer (Host OS) while sharing the same hardware.

3.6.2.2. Virtualization advantages

Virtualization provides several advantages when compared to physical set-ups of hardware (infrastructure) and software (applications). Carroll, Kotzé, and Van der Merwe (2010) conducted a qualitative study on sources regarding the benefits of virtualization. As a result, primary virtualization benefits mentioned by most authors

[7] VMware, Microsoft Virtual PC, Microsoft Hyper-V, Xen, Qemu, KVM, VirtualBox, Parallels, and Bochs.

are a reduction in costs, server consolidation, and server utilization (Carroll et al., 2010). Besides that, some of the stated 'major' benefits are disaster recovery and service continuity (availability), easier or quick deployment, seamless portability and migration, increased flexibility and service agility, reduced downtime, easier and quicker developments and testing, ease of management and administration, isolation, and improved security and control (Carroll et al., 2010).

When looking at the practitioner's domain, the same kind of advantages are found. This domain is directly aimed at businesses, and therefore focusses on briefly stating the advantages organizations can obtain. Next Table 8 denotes these advantages, accompanied with a brief description and corresponding source(s).

Table 8: Virtualization advantages

Concepts	Description	Source(s)
Higher server utilization	By using virtualization, one physical server can now hold multiple virtual machines. Moreover, servers are no longer required to be dedicated to one specific application or service. Virtualization enables servers to divide themselves in order to support different applications. This realizes a higher server utilization than during conventional server use.	NH Learning Solutions, 2017; Carroll et al., 2010.
Reduced hardware costs	As a consequence of the higher server utilization, organizations are not obliged to purchase new servers for each new application. Therefore, reducing the costs for purchasing hardware. The same applies to the maintenance of the servers, since there are less servers that are in need of maintenance.	NH Learning Solutions, 2017; Carroll et al., 2010.
Faster hardware deployment/ Disaster recovery	Virtualization uses images of software clients. Images are back-up files that contain meta-data about a VM or specific software. At the moment a physical server fails, the VM(s) can be transferred to another server, by the means of loading a saved image of that particular VM(s) onto another server. This realizes the exact same VM before the server fail (obviously, it depends on the moment the image is saved).	NH Learning Solutions, 2017; Thrive Operations, 2018; Scroggins, 2013; Carroll et al., 2010.
Energy costs savings	By achieving a higher server utilization, less physical servers are required to possess. This additionally means that less energy is consumed by the organization to run and cool the servers.	NH Learning Solutions, 2017; Thrive Operations, 2018; Tamane, 2015.
Increased staff productivity	Due to the higher server utilization, maintenance staff have more time to focus on other tasks, as the amount of servers is less than during conventional scenarios.	NH Learning Solutions, 2017.
Improved system reliability (incl. redundancy)	VMs are executed in their own isolated environment on servers. The VM is created on the server and operates alone, with no connection to other VMs on the same server. The only connection a VM has is the connection with the hardware (the server), which looks like a computer because of the virtualization layer. Hence, if an error occurs within a VM, the remaining VMs cannot be impacted by this error.	Tamane, 2015; Scroggins, 2013; Carroll et al., 2010.
Increased scalability	Although not all authors mention scalability within the list of advantages, virtualization does increase the ability to scale IT in organizations. Because of virtualization, organizations can change the amount of available applications to different loads, while maintaining the same amount of physical servers. For instance, organizations can temporarily decrease the amount of Service A in order to increase the amount of Services of B, and vice versa.	Thrive Operations, 2018; Carroll et al., 2010.

3.6.2.3. Virtualization disadvantages

Despite all the advantages virtualization brings, the technology also may bring several disadvantages and risks that may affect the overall performance. According to Carroll et al. (2010), most common perceived risks in the virtual environments are regarding security. When looking at the practitioner's domain, security as a disadvantage is not mentioned as it is in the literature. However, several practical issues are denoted as disadvantages. Table 9 briefly elaborates on these risks of both domains.

Table 9: Virtualization disadvantages

Concepts	Description	Source(s)
Security	Security regarding the virtualization environment is influenced by the following factors: fluctuating workloads, dynamic migration and changes, workforce capability and knowledge, access controls, hostile guests, proliferation of VMs, configuration settings, hypervisor and VM monitor layer vulnerabilities, VM server sprawl (complexity).	Carrol et al., 2010.
Complexity of virtual environment	As more applications and services are being executed onto one physical server, the complexity of such an environment is increased. This may lead to a more difficult process of finding an error in the virtual environment (e.g. root-cause analysis).	Pietroforte, n.d.; Sysprobs, n.d.
High risk in physical fault	Due to the server consolidation, a lot more of (sets of) applications and services are being executed on physical servers. A corresponding risk is the consequence of a possible error of such physical servers. When that happens, an amount of applications and services has to be restored onto another server, at the cost of downtime.	Pietroforte, n.d.; Sysprobs, n.d.
Hardware consumption	Each VM gets an amount of hardware resources assigned for its use. In particular, memory (RAM) and disk space is consumed by VMs because of all required resources to run OSs and corresponding applications.	Ganore, 2014;
Performance decreasement	Virtualization does bring an additional layer in the overall architecture to realize a service or application. During operations where speed and a low latency is required, the additional layer may result in a portion of delay. Moreover, virtualization will never be equal to the performance with physical servers in the conventional way. This is, however, not a major issue, as servers without virtualization did not fully utilize their capabilities.	Tamane, 2015; Ganore, 2014; Pietroforte, n.d.; Sysprobs, n.d.

3.6.3. Virtualization and Cloud computing

The cloud domain is enabled by virtualization concepts. Virtualization allows users to access power beyond their own physical IT environment (Harauz, Kaufman & Potter, 2009), meaning infrastructure, and applications and services. Virtualization is therefore regarded as a core component in cloud computing (Carroll et al., 2011). Cloud is defined as "*a large pool of easily usable and accessible virtualized resources (hardware, development platforms and/or services), which can be dynamically reconfigured to adjust to a variable load (scaling), allowing for an optimum resource utilization*" (Vaquero, Rodero-Merino & Caceres, 2009).

Cloud realized the wide introduction of the concepts Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS), as large amounts of computing resources became easily available. Only a sufficient internet connection was required. Conventionally, organizations maintained their own hard- and software in-house, what resulted in a considerable amount of costs. However, because of cloud technologies, organizations could now decide to use one or more 'aaS(s)[8] for their IT environment, and hence, outsource most of the maintenance and configuring, and save on purchases. The different types of virtualization – e.g. storage-, application-, and server virtualization – enabled all different cloud solutions. Figure 24 denotes the enabling role of different virtualization types towards cloud technologies.

[8] ''aaS' is used to denote the collection of *P*aaS, *I*aaS and *S*aaS. I.e. a reference where all three concepts are meant.

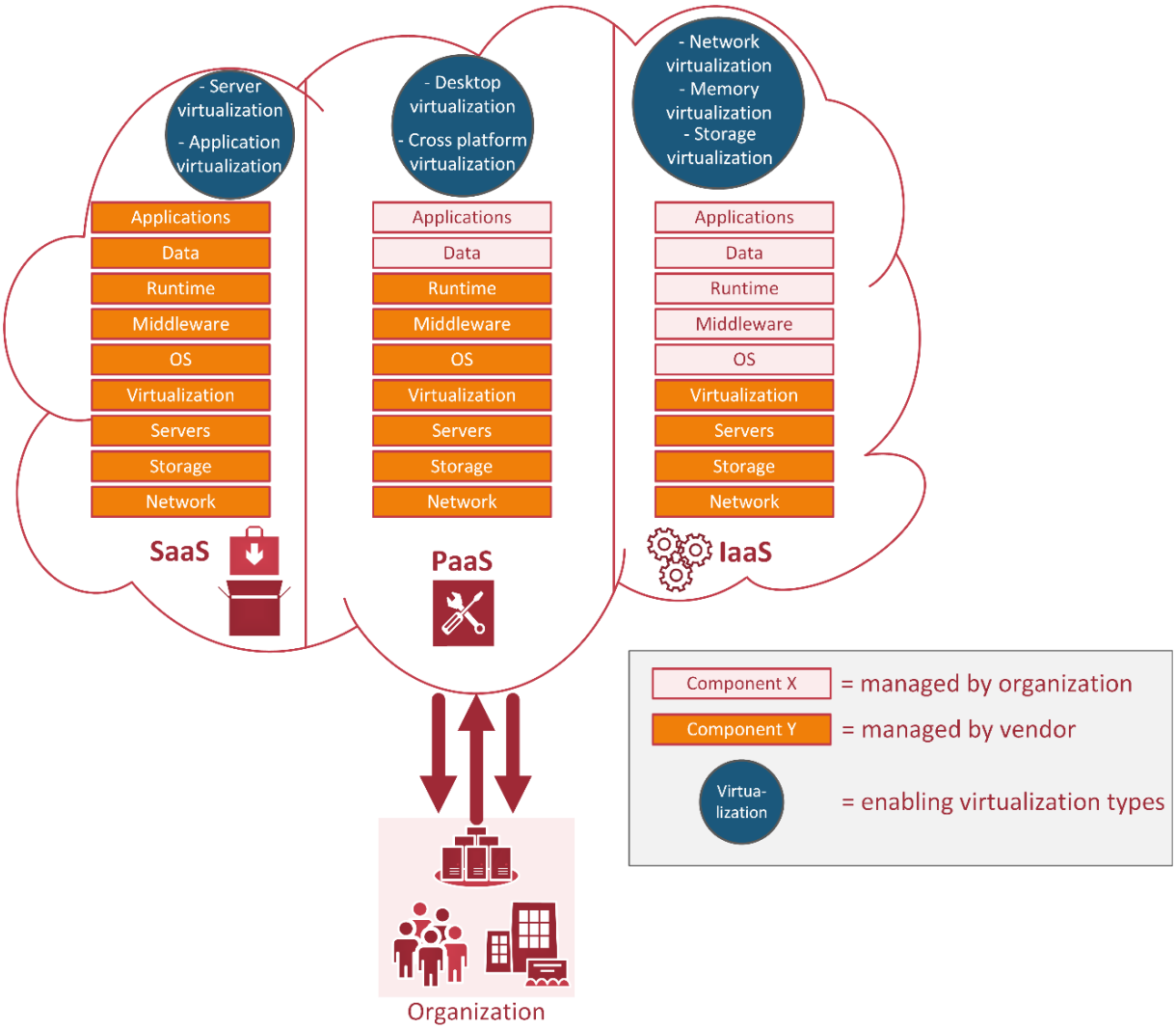


Figure 24: Cloud and enabling virtualization technologies

The increased use of 'aaS realized new business models. This also made a shift recognizable that more organizations chose for this kind of configurations for their IT environment(s). In 2016, 7,3% of all surveyed organizations[9] were using cloud computing, whereas in 2010 only 0,3% of these organizations were using cloud. This is an annualized growth rate of almost 50% (Bloom & Pierri, 2018). Moreover, this shows an increase in the use of 'aaS solutions and, more importantly, it also implies the future role of virtualization technologies.

3.6.4. Virtualization advancement

As a result of the increasing popularity of 'aaS solutions, large data centers exist where multiple virtualization technologies are applied to virtualize physical hardware, and create aggregated pools of computing resources. Organizations can subscribe to these public and private cloud vendors to make use of the offered computing resources in the form of an 'aaS solution. Since there are many vendors with data centers that facilitate different computing resources to organizations through cloud, the next advancement would be to increase efficiency and speed of the virtualization aspects.

As aforementioned, virtualization technologies realized a change in the IT market around 2008. Dawson and Bittman (2008) predicted that virtualization would change the usage of IT, including pricing, application types, and OS usage. Moreover, the authors also stated that there would emerge a competition between vendors that did not exist yet. The SaaS, PaaS, and IaaS concepts of sub-section 2.5.3 are already known results of the

[9] The research of Harvard Business Review is based on the dataset of Aberdeen Information's call center. This data set contains more than 150.000 firms that participated in the survey regarding cloud usage.

utilization of virtualization technologies. Figure 25 depicts in a timeline the change around IT infrastructure inside organizations, and shows the shift that has occurred during the last 50 years. The figure also implies the introduction of a new technology that would further innovate the infrastructure and application domain.

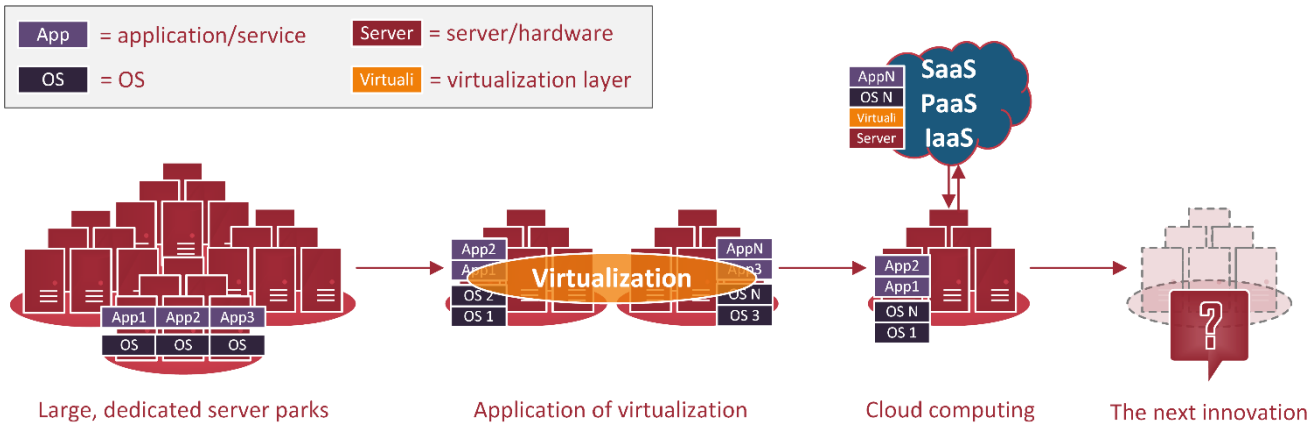


Figure 25: Infrastructure evolution

However, besides the 'aaSs, different vendors attempted to advance the concept of virtualization. An example of such a new vendor is Docker. This organization anticipated on the possibilities of virtualization by developing a concept of containers, popularizing *containerization*.

Moreover, containers are the next innovation in the infrastructure evolution timeline of Figure 26, as containers have particular characteristics. In short, containers provide the same kind of isolated virtual environments as VMs, to enable the execution of applications and services. However, containers are lightweight versions of VMs, meaning that this concept requires less hardware resources. Additionally, containers are launched in a fraction of the time required by VMs. Therefore, containers can be seen as an advanced, increased agile enabling version of current virtualization technologies. Due to this, next section 2.7 elaborates on the container characteristics by providing more in-depth details and differences about containerization.

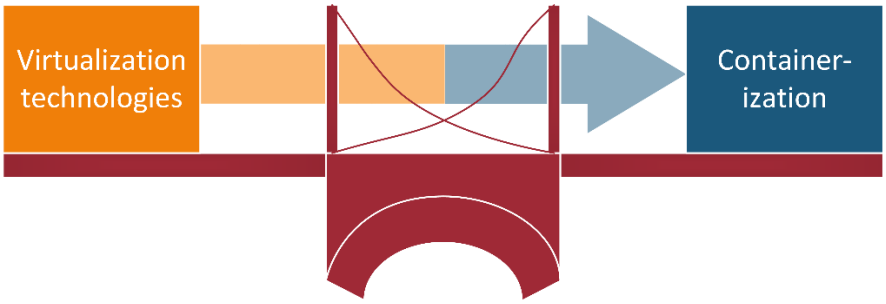
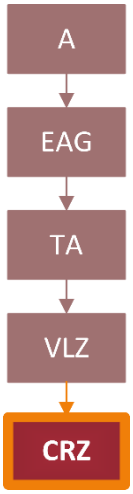


Figure 26: Virtualization technologies linked to Containerization

3.7. Containerization



Containerization is a software concept that has gained popularity since 2013. As mentioned before in section 1.1, more organizations are (planning to) apply(ing) containers in their daily operations. This section describes how containers emerged, what the current state of the art is, compares containers with VMs, and explains how containers are linked to EAG. Throughout this section, containerization and containers are used interchangeably.

3.7.1. History of containers

The origin of containers can be derived from two different needs: 1) the need for isolation (location of files), and 2) the need for a software concept that is independent of the underlying infrastructure.

One of the first solutions that introduced isolation was Unix V7 system in 1979. This system changed the root directory of a process and its children to a new location in the filesystem (Osnat, 2018), resulting in some sort of isolation between parts of the system. Around the 2000s, different hosting providers elaborated on the isolation aspect. Making the smaller systems independently assignable. The first mentioning of 'containers' was in Solaris containers (2004). After that, Google introduced Process containers in 2006, which were designed for limiting, accounting, and isolating resource usage of a collection of processes (Osnat, 2018). Subsequently, Docker launched its containers and container management ecosystem. Docker is used in next sub-sections to explain containers.

The second need was around the independence of a software platform regarding the underlying infrastructure. Java once started as a programming language from the need to have a language that is platform independent. Meaning, the language can be used on different platforms, where it does not matter if it is windows, Unix, or another OS. Containers pursue this generic concept in their functionality. Containers provide functionality that is independent of the underlying infrastructure, meaning the physically enabling and supporting of containers.

3.7.2. Containerization: state of the art

Container-based virtualization and application containerization is an OS-level virtualization method for deploying and running distributed applications without launching an entire VM for each application. Instead, multiple isolated systems (the containers) are running on a single control host (Container Engine in Figure 27) and access a single kernel (Host OS) (Kaur, 2018). Moreover, as opposed to hypervisor-based virtualization (the VMs), container-based virtualization (i.e. OS-level virtualization) is not aimed at emulating an entire hardware environment. It is rather enabling the modern Linux kernel to manage isolation between applications (Li & Kanso, 2015). This means that multiple isolated systems (Jenkins or Salesforce etc.) can run on one control host, sharing a single kernel instance, which is noticeable in Figure 24. Each container having its own process and network space realizes this multiple, isolated concept (section 2.7.2.1 elaborates on this). In essence, a container is a set of processes (usually with storage associated) completely isolated from other containers (Li & Kanso, 2015).

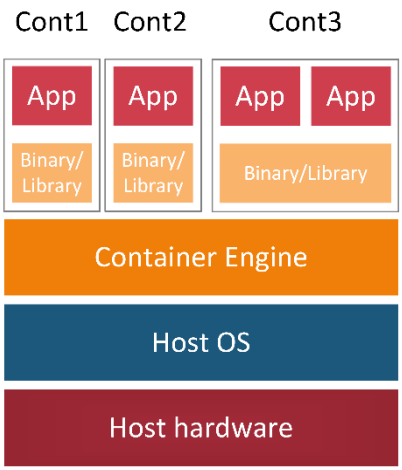


Figure 27: Container architecture high level

However, other differentiating descriptions about containers exist. According to Firesmith (2017), a container is a virtual runtime environment that runs on top of a single OS kernel and emulates an OS, rather than the underlying hardware. Another example is Pahl's (2015) description of containers: "*a container holds packaged, self-contained, ready-to-deploy parts of applications and, if necessary, middleware and business logic (in binaries and libraries) to run applications*". Besides that, most authors also mention that containers, in comparison to VMs, are lightweight and therefore realize a lower overhead, provide fast start-up times, and bring the capability to interconnect containers and rapidly scale large amounts (Amaral et al., 2015; Pahl, 2015).

The container domain consists of two types of technologies: 1) container solutions and 2) container orchestrators (see subsection 2.6.4). Container solutions provide the same type of container services (deploy, manage, and scale applications), but differentiate from each other when going into details. In reality, each solution can be used to solve different things and are rooted in varying contexts (Abdelrazik, 2017). This is due to their individual specializations, applied techniques, and their own goals. An overview of most used vendors and their specializations is provided in Appendix IV – Containerization vendors.

Figure 28 depicts the characteristics of containers that are the most found during the literature review.

Figure 28: Container characteristics (based on all sources used in section 2.3.2)

3.7.2.1. Container architecture

The container architecture comprises of multiple layers. These layers facilitate the transition between the host OS and the container engines, and different types of desired software. Figure 29 depicts the container architecture, and has the same structure as Figure 27. The architecture of a Linux container is chosen to depict, since most containers applied in practice are Linux-based.

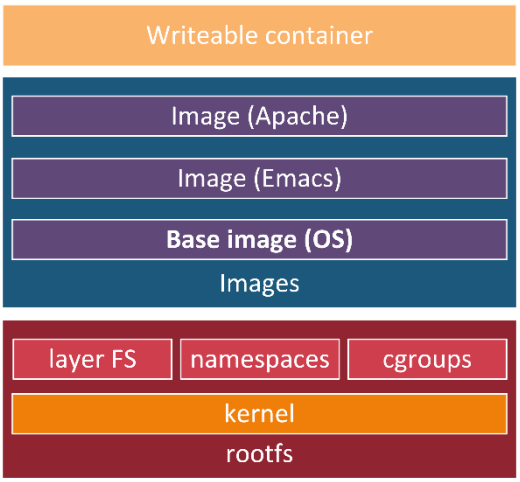


Figure 29: Linux container architecture – Pahl (2015) (modified)

The general architecture consists of three main layers:

- Writeable container;
- Images (of OS and other software types);
- Rootfs (file system of the root).

Layer 1: rootfs

[10]Rootfs is connected to the physical host hardware, and comprises of the host OS' kernel (Windows or Linux). The kernel mounts the filesystem of the host OS (read-only), and subsequently adds another filesystem on top of it (writeable).

Above the kernel, three concepts are displayed that provide different functionality. Layer FS is a filesystem layer where the above components (images and writeable containers) can derive data. For each new image in layer 2, an individual FS layer is launched.

Besides that, different OS distributions (e.g. Linux) provide kernel mechanisms such as namespaces and cgroups to isolate processes on a shared OS (Pahl, 2015).

Namespaces isolate the set of filesystem mount points seen by the group of processes (Amaral et al., 2015). In other words, this concept prevents groups of processes from resources in other groups. Container technologies use different namespaces for all kinds of goals, such as process isolation, network interfaces, access to inter-process communication, and for isolating kernel and version identifiers (Pahl, 2015).

Cgroups (control groups) organize the processes in a hierarchy tree. Additionally, cgroups manage, limit, and account the resource access of containers/process groups by setting up resources limits and constraints, and by allowing containers to share hardware resources (Amaral et al., 2015; Pahl, 2015).

Layer 2: Images

The second layer comprises of two types of images. An image is a serialized copy of a software type (system, process, database etc.) stored in one file. The Base image is a copy of an OS that is going to be used for the

[10] Description of the container architecture is bottom-up.

container. The other images are additional services that can be executed in the container. For example, besides Linux OS, Apache service is also launched in order to run an application that requires local Apache support.

Layer 3: Writeable container

The top layer is the container itself, which is the only layer that is writeable. This means that the container can have a state and is executable. The container can be seen as a directory that contains everything needed for execution (Pahl, 2015). Besides that, containers can be processed into stateless images (see next paragraph) and reused in more complex builds (i.e. clustering groups of containers). Section 2.7.4 Container orchestration elaborates on this complexity.

3.7.2.2. Container advantages

Containers have multiple pros in comparison to conventional VMs. Table 10 provides an overview of these advantages. However, the main advantages of containers are lightweight, enabling flexibility, and being launched faster than conventional VMs and at the cost of fewer resources.

Table 10: Container advantages

Concepts	Description	Source(s)
Hardware costs/ server utilization	Container-based virtualization further enables allocation of multiple applications to the same hardware, improving hardware utilization and resulting in decreases hardware costs.	Firesmith, 2017; Flow.ci, 2016; Showell, 2015.
Scalability	A single container engine and host OS can manage large numbers of containers, and rapidly scale both up and down when required.	Firesmith, 2017.
Spatial isolation	Containers support lightweight spatial isolation by providing each container with its own computing resources and container-specific namespaces.	Firesmith, 2017.
Storage	Containers are lightweight in their storage size when compared to VMs. One of the reasons is the unrequired Guest OS. Additionally, applications within containers share the same binaries and libraries.	Firesmith, 2017; Flow.ci, 2016; Showell, 2015.
Performance and speed	In comparison to VMs, containers increase performance (also denoted as throughput) since containers do not emulate the underlying hardware by using hypervisors, but operate directly on the hardware through the container engine. Due to this, containers have a greater contribute to enabling real-time applications.	Firesmith, 2017; Flow.ci, 2016; Showell (2015).
Continuous integration	Containers support agile and continuous development processes by enabling the integration of increments of container-hosted functionality. The aforementioned main advantages also enable this.	Firesmith, 2017; Flow.ci, 2016; Showell, 2015.
Portability	Portability from development to production environments are supported by containers. Especially cloud-based applications have benefits by using containers.	Firesmith, 2017.
Security and new versions	Modular architecture provided by containers increases the complexity and difficulty of attacks. When one container is infected, attacked, or hacked, the impact is mostly limited to the impacted container due to isolation. That container can be destroyed and replaced with a new container, which is loaded on a clean and known image. This enables rapid system restore or software reload following a cybersecurity compromise. In addition, security software and rules implemented at the container engine can apply to all of its containers.	Firesmith, 2017.

3.7.2.3. Container disadvantages

However, containers also consist of several disadvantages when compared to conventional VMs. The following Table 11 denotes the disadvantages as perceived by the field of practitioners. It must be denoted that according to the expert interviews, the biggest disadvantages of containers is for organizations the lack of knowledge. Besides that, concepts of table 11 are perceived as disadvantages, but do not have major impact when applied in practice.

Table 11: Container disadvantages

Concepts	Description	Source(s)
Shared resources/ reduced isolation	Applications within containers share many resources, including container-specific resources (container engine, host OS, kernel, binaries and libraries, networks and host hardware). This sharing of resources can result in a single point of failure, and software running in container A can impact software running on container B.	Bigelow (n.d.); Firesmith, 2017.
Interference analysis / networking	Shared resources imply interference. Due to multiple containers and applications, the difficulty of interference analysis is increased and more complex. Interference paths are getting more complex, as many components are a shared resource.	Firesmith, 2017.
Security	Containers are not immediately secure. In order to make them secure, it must be ensured that: 1) zero data is stored inside containers, 2) container processes are forced to write to container-specific file systems (designated storage), 3) container's network (namespace) is connected to a specific, private intranet, and 4) the privileges of container services are minimized (e.g. non-root access).	Firesmith, 2017; Flow.ci (medium.com); Showell, 2015.
Container sprawl	Excessive containerization is relatively common and increases the amount of time and effort spent on, and knowledge and skill required for container management.	Bigelow (n.d.); Firesmith, 2017; Showell, 2015.
Less OS flexibility	Since containers share the same OS as the host OS, starting-up a new server is required if one wants to use another OS in a container. Although this is not an issue for large hosting parties.	Firesmith, 2017; Flow.ci (medium.com); Showell, 2015.

3.7.3. Containers and stateless and stateful applications

The concepts of stateful and stateless applications and servers have a roll in the execution of containers. According to Comer and Stevens (1993), a server that maintains information (state information) about the status of ongoing interactions with clients is a stateful server. In contrast, servers that do not keep state information are stateless servers. Briefly explaining, a stateless server always accepts the same set of requests from clients and reacts only dependently on the commands given (Perrochon, 1994). A stateful server, however, has different states. This means that this server can accepts/denies information, based on its state. In other words, with stateful servers, the generated output is dependent of the input and the current state of the server, whereas in stateless servers, the output is solely dependable of the input.

Containers also have a certain state, as there can be multiple instances of a container (v1.0-v1.1-v1.2 etc.). Nonetheless, most of the times containers are used as stateless environments where applications and services are running. This means that there is no data stored in a container across the service's operations. Once a container is destroyed, the data is vanished. Being stateless enables services and applications to retrieve an application's state from another storage, and exploit it to execute according to a certain state behavior (as depicted in Figure 30). This contributes to realizing the flexibility and scalability benefits of containers. However, currently practitioners are experimenting with stateful containers, meaning that the data is persistent instead of vanishing. Examples are containerized databases and servers with mailing data or transactional data.

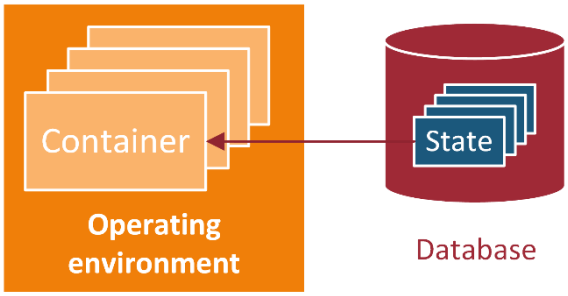


Figure 30: Stateless containers deriving states

3.7.4. Container orchestration

Container orchestration is the deployment and management of multiple (clusters of) containers. According to Pahl, Brogi, and Soldani (2017) orchestration is defined as "*constructing and continuously managing possibly distributed clusters of container-based software applications*". It allows users to define how to coordinate containers in the cloud when deploying a multi-container application, and to manage multi-containers as a single entity.

Container orchestrators have been developed as these engines have functionality to manage issues that conventional container technologies were not able to manage:

1. Built-in mechanism to restart any failed containers;
2. Tied to multiple hosts, which is easily distributable and scalable to multiple instances;
3. Simple and summarized information of the script, instead of extensive lines of code (Hohn, 2017).

Additionally, containers can have multiple instances that emerge and disappear. Hence, orchestrators must be able to handle this by automatically giving a unique name to each container instance. Besides that, orchestration tools need to interact with stacks of an IT infrastructure from cloud services to data center resources. Therefore, orchestrators need scalable infrastructures that allow dynamic allocation of resources to enable deployment of applications (Tosatto, Ruiu & Attanasio, 2015).

3.7.4.1. Container orchestration architecture

Containers are grouped into host nodes. Such host nodes are virtual servers on hypervisors or bare-metal servers. Each host holds several containers with applications and services (Pahl & Lee, 2015). From a higher level, a container-based architecture consist of multiple hosts, which are grouped into clusters. The containers in a cluster hold different services or applications, which can be interrelated to each other, or individual. Next Figure 31 depicts an example of a container-based cluster architecture. Kubernetes is also added to this model, as it is the most used orchestration tool in the field of practice. In addition, the external operational environment is also given to complement the interaction of the overall architecture.

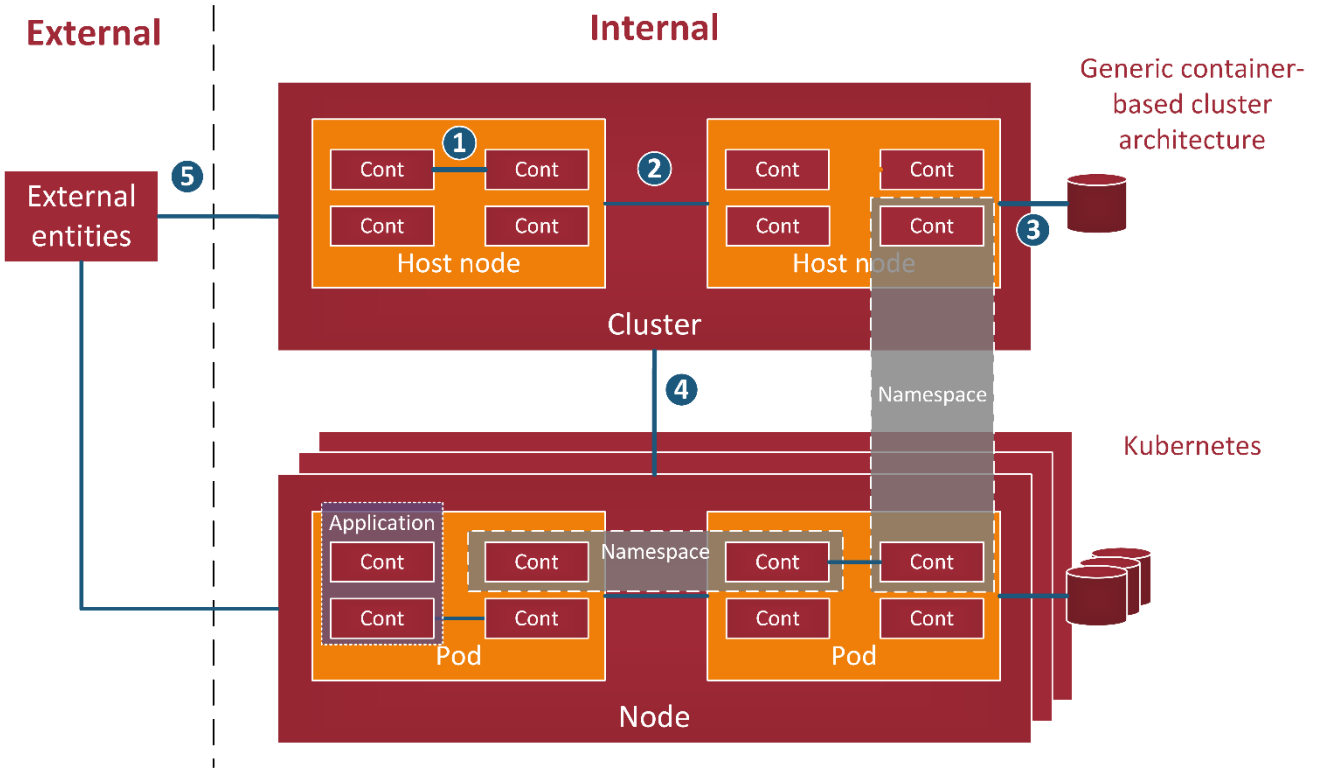


Figure 31: Container-based cluster architecture - based on Pahl and Lee (2015) (extended)

In the container-based cluster architecture of Figure 31, different lines of connectivity (blue) are shown. The first (1) denotes the connectivity between two or more containers. This means that these containers share the same kernel or have the same image. Next line (2) depicts the connectivity between different host nodes. For instance, services originating from Host A are used in Host B. The third line (3), shows the connectivity between a host node and its containers to a database. This database consists of stateful data, which is derived and used in the containers' services. The connectivity between the cluster and node (4) shows the required relation between clusters that enable scalability. Finally, the fifth line (5) denotes the connectivity between an

organization's infrastructure and external entities (third party infrastructure, e.g. cloud vendors and clients). In addition, an example of containers that enable a specific application is depicted in purple. The dark grey areas depict namespaces, which are networks of virtual group(s) of containers, or nodes/pods. Finally, Kubernetes handles an additional upper layer called Cluster (light-orange), which groups multiple nodes. This is because of the detailed architecture of Kubernetes. Further elaboration on this detailed architecture is not provided, as it is not in scope of this project.

This architectural scenario is an abstraction layer for cluster-based service management (Pahl & Lee, 2015), and is hence generally used as architecture within orchestration of containers. Such cluster management architecture enables the deployment of distributed applications through containers. Orchestration tools manage these kinds of architecture, and use their specific jargon.

Table 12: Cluster management architecture components (Pahl, 2017) (modified)

Concepts	Description
Service node (cluster)	Realizes the services and applications of containers. Contains of high internal complexity, but low external complexity, meaning that these clusters are easily to manage by tools.
API	Allows operating clusters from the creation of services and containers sets to other life-cycle functions.
Platform service manager	Handles the software packaging and management.
Lifecycle management agent	Manages container lifecycles at each host.
Cluster head node service	The master that receives commands from the outside and relays them to container hosts.

These components enable the aforementioned benefits of container-based architectures. Especially cloud technologies benefit of these scaling, availability and networking resources managing. Moreover, next to virtualization being the core component of cloud computing, Bernstein (2014) adds containers as the other core element of cloud computing nowadays. In 2014, several public cloud platforms were already using containers instead of VMs. Examples are Google, IBM/Softlayer, and Joyent (Bernstein, 2014).

3.8. Differences between VMs and containers

As we described VMs and containers in the prior sections, this section elaborates on the differences between both concepts. First sub-section provides an explanation about the high-level differences between VMs and containers. The follow-up sub-sections focus on more detailed container advantages and disadvantages in comparison to VMs. With these sub-sections, we aim to prove based on scientific sources that containers are the preferred virtualization technology to apply in the application and infrastructure landscapes of nowadays.

3.8.1. High-level differences

"*Although VMs and containers are both virtualization techniques, they solve different problems*" (Pahl, 2015). Containers share the same OS kernel as their host, which realizes an increase in the efficiency when compared to VMs. This is because VMs require for each VM a full, individual OS (the Guest OS) and additional binaries and libraries to be able to launch an application. These full OS packages create a space concern regarding computing power resources and disk storage, and may lead to several minutes of starting-up a VM (Pahl, 2015). This is the reason that containers are perceived as lightweight, enable rapid scalability, and reduce overhead costs. Figure 32 depicts the difference between both virtualization concepts through a high-level architectural view.

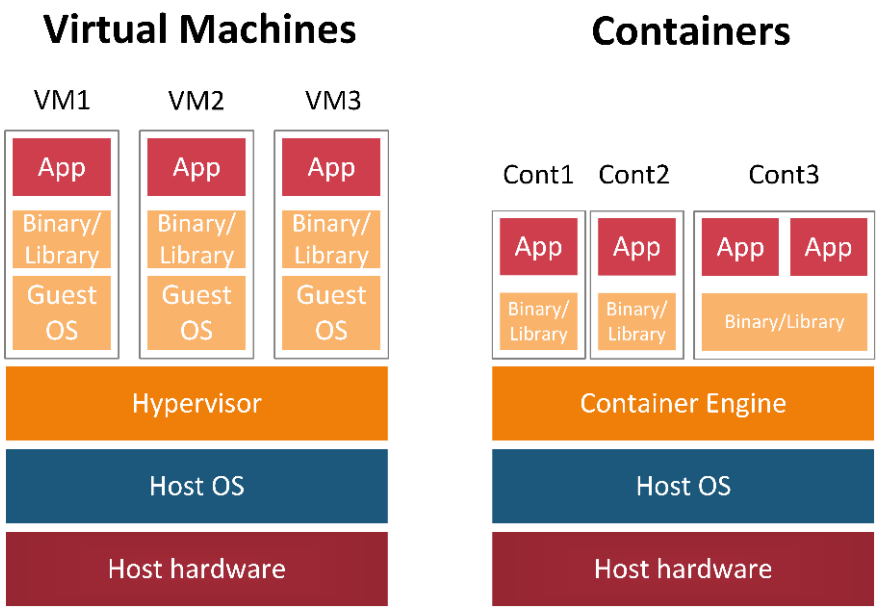


Figure 32: VMs vs Containers

The figure shows the aforementioned characteristics of both virtualization technologies. The main difference is that a VM emulates hardware, while a container emulates an OS (Firesmith, 2017). Moreover, containers use a small part of the Host OS to provide its applications and services. Besides that, containers already contain the components required to run the desired software, such as files, environment variables, dependencies, and libraries (Kaur, 2018).

Regarding infrastructure, when a VM is launched, the assigned hardware is emulated to the machine. In this way, the VM is restricted to solely use the available computing resources. Containers, however, follow a different method to divide infrastructure among them. This management of a container is performed by the Host OS (section 3.7.2). Finally, containers have increased performance in comparison to VMs, since containers reach near native speed in processing, memory, and network throughput while using its computing resources (Higgins, Holmes & Venters, 2015).

3.8.2. Container advantages

Both scientist and practitioners know the advantages of containers in comparison to VMs. Due to the wide usage of virtualization by organizations, (IT) scientists started to compare performance of virtualization and containerization (i.e. VMs versus containers). For instance, Dua, Raja, and Kakadia (2014) measured the

performance of VMs and containers for the support of PaaS. The authors stated that containers have an "*inherent advantage*" over VMs due to increased performance and reduced startup time. Besides that, the same is concluded in a study by Seo, Hwang, and Moon et al. (2014), whom tested performance of both containers and VMs in building cloud environments. Moreover, Sampathkumar (2013) argued that "*Linux Containers provide a significant advantage by performing the common tasks of provisioning, booting, and rebooting a virtual machine several times faster than"* VM-based solutions, emphasizing the same statement. Additionally, a study executed by IBM tested performance of containers in the context of cloud computing. The authors stated that containers result in equal or better performance than VMs in most of the cases (Felter et al., 2014). Joy (2015) also found that containers require "*very much less*" time to scale and process the requested service than VMs require, implying higher performance.

Sampathkumar (2013) mentioned in the same article that Linux Containers exhibit the least overhead in virtualizing CPU and disk access, denoting the near bare-metal performance advantage. The argument concerning overhead is also emphasized by Scheepers (2014), Joy (2015), Seo et al. (2014), Xavier et al. (2012), and Felter, Ferreira, Rajamony et al. (2014).

3.8.3. Container disadvantages

On the other hand, most studies also discussed several negative aspects of containers. Scheepers (2014) compared containers with VMs on application infrastructure. This author denoted that realizing the advantages of containers is context-specific. The advantage of fewer overhead and near bare-metal performance of containers come at the cost of poor isolation of computing resources in comparison to VM solutions (Scheepers, 2014). The aforementioned publication of Dua et al. (2014) additionally mentions that containers have a "*bright future*" when there is more standardization and abstraction at the kernel and Host OS. With this, the authors imply the same statement about the current state of isolation as Scheepers (2014). Moreover, Sampathkumar (2013) also embraces this statement as well as Xavier et al. (2012) and Joy (2015).

Primarily, we hypothesize that this is due to the fact that VMs have more mature characteristics – i.e. further developed features that improve quality of functionality – than containers do. Li and Kanso (2015) make the same kind of statement. In their paper, the authors argued that container-based environments are missing mature features that conventional VMs do possess. Moreover, Xavier et al. (2012) also concluded that container-based systems are not mature yet, based on the examination of isolation results. In addition, Felter et al. (2014) argues as well that containers are only recently adopted and standardized for the use of mainstream OSs, resulting in a "renaissance" in the use of containers. In other words, containerization is still an immature concept.

3.8.4. Differences and similarities rationalized

However, most of those publications originate from the first half of current decade (2010-2015). As aforementioned in section 1.1 Problem statement, in 2013 Docker was launched what popularized containerization. As global usage of containers increased during those years, it is evident that containers were in development by both vendors as (software) researchers. This resulted in containers becoming (more) mature, while gaining mature features over the years. Examples of such developments are the introduction of easy to configure container management tools (e.g. Kubernetes), Ceph and REX-Ray focusing on container-based storage, and tools like Jenkins to foster SDLC (container-based application development) (Osnat, 2018).

More recently, Tesfatsion, Klein, and Tordsson (2018) published an article where they compared performance of multiple virtualization technologies (hypervisor-based and container-based) in clouds. The authors concluded that containers have a higher advantage in scenarios where multiple instances of applications or services are to be executed without strict performance requirements (e.g. cloud providers).

Based on all of the aforementioned sources of sub-section in section 3.8, we conclude that containers prevail above VMs in terms of performance, resources usage, efficiency, and portability. Although, despite the development of containers throughout the years, containers still experience disadvantages when compared to VMs. Main example is reduced isolation capabilities due to the shared kernel concept.

Several authors argue that security of containers can be improved. For instance, Tesfatsion et al. (2018) stated that it is possible to decrease the security (i.e. isolation) issues of containers by extending existing security policies. The authors additionally mention *unikernels*, which should offer better security and efficiency than traditional OSs. Moreover, Arnautov, Trach, and Gregor et al. (2016) claimed to have improved container security by using Intel Security SGX (Software Guard Extensions). Besides that, Bui (2014) concluded that security levels of containers could be increased by applying several actions or additional extensions. Further elaboration onto this topic is out of scope for this project, as security is currently not a crucial topic for the development of the CMM. However, as containers are in development regarding their security, we think it is important to consider these developments in a later stage. Hence, we mention this topic in Future work.

Concluding, we rationalize that receiving benefits of containers depends on the use case. For instance, the use case of rapid deployment (Sharma, Chaufournier, and Shenoy et al., 2016) is in our opinion container-fit. This means that containers are also relevant for the SDLC process, as this process is benefitted by rapid deployments and high portability. Moreover, this argument is emphasized by Sharma et al. (2016), as the authors state that containers enable improvements in SDLC.
From a global perspective, containers beat VMs in overall performance. The important distinction of both virtualization technologies is the in-depth architecture and the resulting difference of functionality. When a virtual environment is handling sensitive information or should be able to hold data for a period of time (stateful), a VM should be selected due to its isolation benefits and persistence. Lastly, both Tesfatsion et al. (2018) and Sharma et al. (2016) mention the potential of hybrid solutions. This means combining VMs and containers with each other to enable the architectural advantages of both.

Finally, in this section, we provided a theoretical explanation about containerization and virtualization. However, we wanted to denote the difference between both concepts with functional examples from practice. Therefore, we created a table where we stated main EAG attributes of Sherehiy et al. (2007) and incorporated both the container technology and VM benefits.Through this table, we intended to denote the differences in support of the core agility attributes. This table (table 2) is positioned in section 1.1.4 on page 14.)

3.9. Software Development Life Cycle

Software Development Lifecycle (SDLC) is the complete lifecycle of a (piece of) software. Software can be a single service, an application, a complete system etc. A lifecycle covers all the stages of software from its inception with requirements definition through to the release and maintenance (Ruperalia, 2010). According to Ruperalia, SDLC is a conceptual framework or a (set of) process(es) that considers the structure of stages involved in the development of an application from its initial feasibility study through to its deployment in the field and maintenance. Different SDLC models exist where two main groups can be identified. The first one is *Traditional software development* methods (e.g. Waterfall, V-model), and *Agile software development* methods (e.g. Rapid Application Development (RAD), Extreme Programming (XP), Scrum). Inside the two groups, three types of models can be distinguished: 1) Traditional is primarily characterized as linear, whereas 2) Agile is characterized as iterative. The third type is a combination between both (Leau, Loo, Tham et al., 2012; Ruperalia, 2010).

3.9.1. The SDLC phases

In general, all SDLC models follow the same kind of process steps. This includes the phases as shown in table 13.

Table 13: SDLC phases and descriptions

#	SDLC phase	Description	Involved components
1	**Code**	All tasks involved that developers perform for producing lines of code.	• Source code management system (i.e. Code repository); • Base images.
2	**Build**	Build is the phase that all written (parts of) code are formed together and are being built into an executable version. Different kind of pre-building checks are performed to determine if the build is going to be successful. Builds can be in different granularity levels: code-level and functionality-level (set(s) of code).	• Artefact repository; • Build servers; • Build controls.
3	**Deploy**	This phase deploys the build of a (piece of) software to a desired configuration-specific (virtual) environment. In such an environment, the build (code) will be tested. This phase is iterative nature with the test phase.	• Containers; • VMs; • Deploy states
4	**Test**	Different types of tests are testing all deployed builds. If a test fails, the code has to be improved, rebuild, and redeployed in order to restart testing. These tests differ from basic, rather simple compile and unit testing, to more complex exploratory and performance testing.	• Containers; • VMs; • Test/acceptance environment.
5	**Release**	When all tests are successfully completed, the (new version of) application is released into production. This means that the application is running and end-users are using it.	• Containers; • VMs; • Release controls.

3.9.2. SDLC and virtualization technologies

VMs and containers can both be applied for equivalent tasks and goals in the SDLC process. Overall, virtualization technologies are applicable to SDLC in two different levels:

- **Process level**: throughout the phases of the SDLC process during its execution;
- **App & Infra level**: as a support from the application landscape and infrastructure for enabling the SDLC process by provisioning the required applications, services, and hardware instances.

Referring to the SDLC process level, virtualization's main application is in the Deploy and Test phase (orange stars in Figure 33). In here, virtualization launches virtual environments where software is tested in. Regarding the support level, virtualization is used in both the application landscape and infrastructure domain. A visualization about both levels is provided in the figure below.

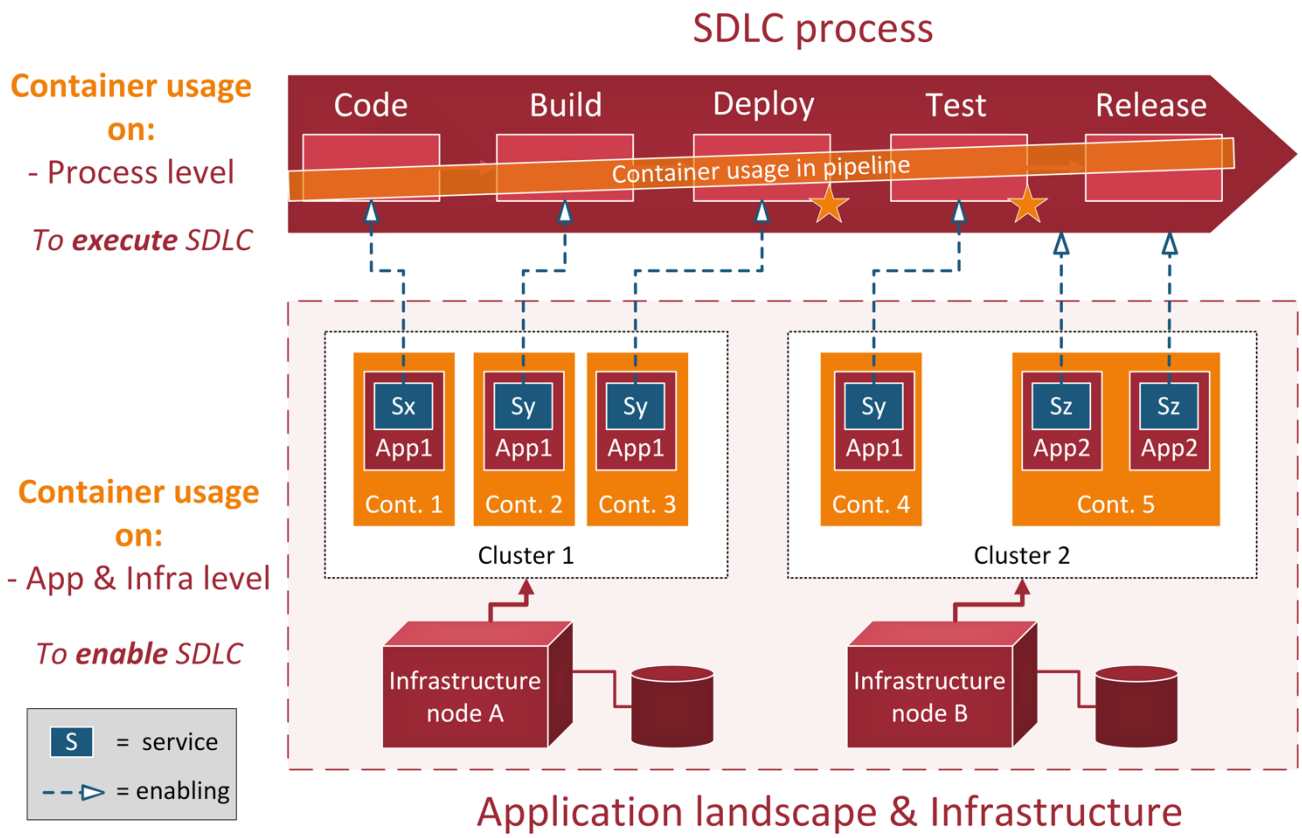


Figure 33: Container usage on process and architectural level

As the figure depicts, each phase uses a service to support the execution of the process. Figure 33 shows a simplified situation (each phase uses one service), but in practice a larger amount of services is used by each phase (see section 3.8.2). This means that the SDLC process – interchangeably denoted as 'software delivery lifecycle', 'software delivery pipeline' and 'software development pipeline' – can be seen as a cycle that uses and delivers software. Based on our research, we came to the following distinction for our understanding:

- **Software development pipeline** refers to the process of actual development of (new versions of) software.
- **Software delivery pipeline** refers to the launching of new instances of existing software, in order to support or provision SDLC in both contexts (and other processes in an organization).

3.9.3. Tooling and applications

As aforementioned, different tools and applications are used throughout the SDLC process. Tools are designed that focus on one or more SDLC phases. For instance, tools exist that are designed specifically to enable and support CICD. On the other hand, there are specialized tools that focus on one of the SDLC phases through code reviewing, testing (e.g. unit- or API-testing), deploying containers and monitoring. Moreover, large IT organizations such as Google or Amazon offer tools that contain complete delivery pipelines.

As shown in Figure 33, containers are applicable at different locations in an organization. The SDLC pipeline uses multiple services with varying demands in capacity. Currently, VMs are still leading in usage, whereas containers are better suited for these kind of (non-persistence) tasks. This means that on all locations VMs are used, containers are also applicable.

3.10. Maturity (assessment) models

In the final decades of last century, the software industry had a need for solutions regarding software quality issues. One of the answers were maturity assessment models. The concept of *maturity* was popularized through the introduction of the Capability Maturity Model for software (CMM-SW), which was developed by the Software Engineering Institute (SEI) between 1986 and 1993 (Schlichter, McEver & Hayes, 2010). Maturity models were developed from the organizational need of software improvements, which led to the domain of Software Process Improvement (SPI). Results are subsequently used to denote areas of improvement. Existing maturity models are ISO SPICE, PRISMS, OPM3, N2C2M2, P3M3, and the more recently variant of CMM: CMMI.

Multiple software development processes exist in organizations, and there are different sequences of tasks, tools, and techniques available to plan and implement improvement activities. Due to this wide selection of process improvement scenarios, Paulk et al. (1993) found that it is difficult to achieve consensus between management and the professional staff on what improvements activities to undertake first. The authors argue that organizations should follow a structured, evolutionary path that increases an organization's software process maturity in multiple stages. These stages are the so-called maturity levels.

3.10.1. CMM(I)

According to Paulk et al. (1993), a *maturity level* is "*a well-defined evolutionary plateau toward achieving a mature software process*". Each maturity model consists of multiple maturity levels, and each level denotes different process improvement topics for that specific plateau. Due to the aforementioned popularity of CMM, the general structure of CMM(I)[11] is also widely adopted by other maturity models. Hence, we use this specific structure to further elaborate on maturity models. Figure 34 depicts the structure of all elements of CMMI.

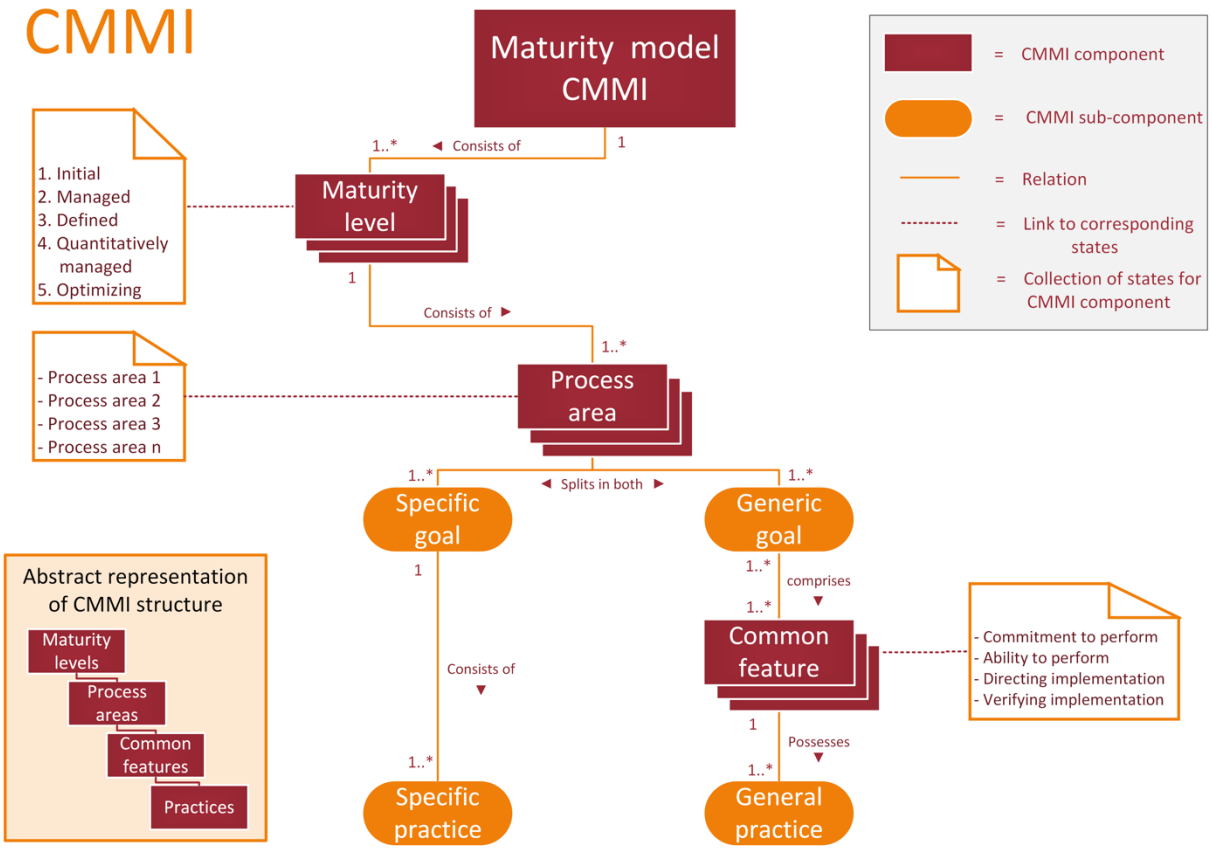


Figure 34: CMMI model structure

[11] CMMI is the successor of CMM. The major difference is the addition of 'Integration', meaning that the software process is further integrated throughout the whole organization. This can also be noticed in the process areas. Whereas the key process areas of CMM are primarily focused on software, the equivalent process areas of CMMI have a more organizational focus. This enterprise-widening focus of the model aligns with EAG.

In the CMMI model, each maturity level consists of a set of process goals that stabilize an important component of the software process. Achieving each level of the maturity framework establishes a different component in the software process, resulting in an increase in the process capability of the organization (Paulk et al., 1993). The CMMI structure exists of the following levels:

1. **Initial:** Software process is characterized as ad hoc, and sometimes acts as a black box. Few processes are defined, and success depends on individual effort.
2. **Managed:** Basis project management processes are established to track cost, schedule, and functionality. Projects ensure that requirements are managed, and processes are planned, performed, measured, and controlled.
3. **Defined:** Software processes for both management and engineering activities are formally documented and understood, and described in standards, procedures, tools, and methods for the organization.
4. **Quantitatively Managed**: quantitative objectives for quality and process performance are established and used as criteria in managing processes. These objectives are based on needs of customer, end users, organization, and process implementers.
5. **Optimizing:** focused on continually improving process performance through both incremental and innovative technological improvements.

The levels show a growth process, starting with a set of black box software development processes (level 1), to structured and formally documented software development processes while continuously improving (level 5).

3.10.2. OPM3

Besides CMMI, another maturity model named OPM3 (Organization Project Management), handles a different hierarchy of applying maturity in organizations. At first, a set of processes have to be improved for Projects. Subsequently, these process improvements should be applied to Programs, followed by Portfolios. This shows that an organization can also mature by incrementally improving from the lower domains to higher domains (*hierarchy levels)*. This scenario is depicted in Figure 35.

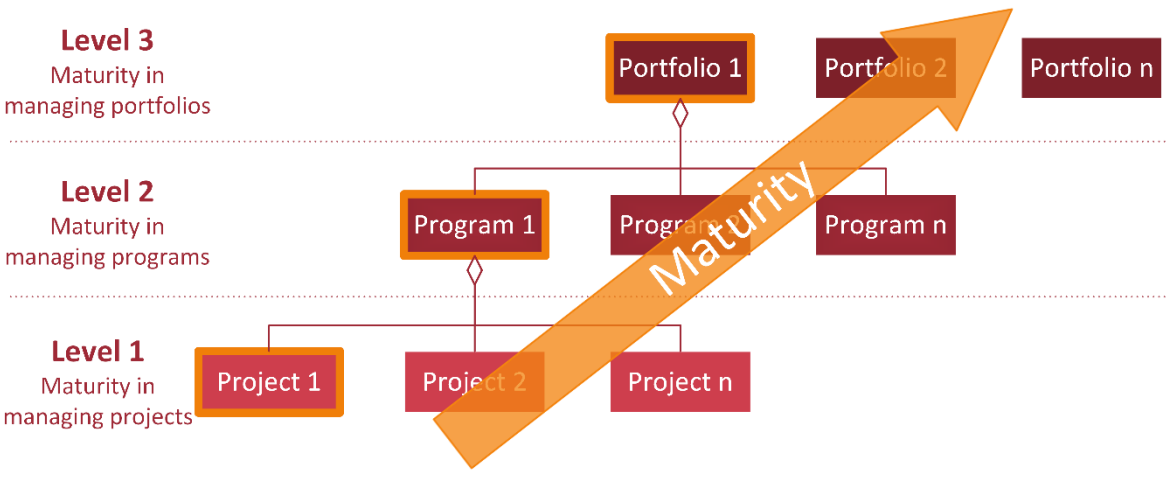


Figure 35: maturity applied at different project management domains

However, these hierarchy levels are goals to increase maturity according to OPM3. The cycle of how maturity is gained is divided in four levels, which are 1) Standardize, 2) Measure, 3) Control, and 4) Improve (Project Management Institute, 2008). From an abstract perspective, the same growth structure as in CMMI can be perceived. At first, basic management processes are documented, what enables standardization. Subsequently, more detailed aspects are identified, which results in measurable indicators. After that, process performance and product quality is monitored using these indicators. Finally, all collected information is used to (continuously) improve the software process.

When comparing other maturity models, we notice the same structure as defined in CMMI. This shows that a recurring pattern can be distinguished from all these models. This pattern will be used during the design of the proposed container maturity assessment.

3.10.3. Dreyfus Model

The five-stage model of adult skill acquisition of Dreyfus and Dreyfus (1980) (A.K.A. Dreyfus model), and is a model that describes the learning process of an individual that wants to learn a new skill. Different versions of the model exist, which use different names for denoting maturity levels. We use the version of 2004, as we think these names correspond more with our definition of the maturity levels. Four mental functions are used in the model: Recollection, Recognition, Decision, and Awareness. Each function is defined according to a certain behavior at each maturity level. The maturity levels are the following (derived from revision of Dreyfus, 2004):

- **Novice**: context-free features that beginners may recognize without possessing the desired skill. Beginners are provided with basic rules about the skill.
- **Advanced Beginner**: novice begins to start understanding the skill of relevant context, as novice experienced real situations. Additionally, more advanced rules and aspects of skill are becoming recognized.
- **Competence**: the learner is recognizing more procedures, relevant elements, and rules. Performance is starting to decrease, as the all information can be overwhelming what makes it difficult to correctly prioritize.
- **Proficiency**: at this stage, the learner's "*theory of the skill, as represented by rules and principles, will gradually be replaced by situational discriminations*" (Dreyfus, 2004). Goals and important aspects are recognized, however required actions to achieve these are not due to a lack of experience.
- **Expertise**: the performer sees the goals and now additionally knows how to achieve these goals. Also, the ability to make subtler and refined distinctions between goals and tasks.

Equivalent to CMMI, the Dreyfus model shows a path of growth in its maturity model. However, there is an important difference between the models that implies a certain expectation, which may lead to confusion.

Dreyfus denotes names for the human learners of a skill and describes expected, general behavior per maturity level as levels of skill. Rather than the CMMI model, where the maturity level names lean more towards functional requisites. For example, Level 2 Managed implies that a process is managed to a certain extent, and level 4 Quantitatively Managed even involves quantification of the process to be able to start measuring its performance. This corresponds with the highest maturity level – Optimized – that focusses on continuously optimizing.
This implied functionality with the focused on processes, restricts adopters of this model structure to a direction of maturity, instead of describing behavior without limiting model adopters.

3.10.4. Software process maturity

As we focus on maturity of the SDLC process of organizations, software process maturity is briefly investigated. According to Paulk, Weber, Garcia et al. (1993), a **software process** is defined as a set of activities, methods, practices, and transformations that people use to develop and maintain software and associated products. Examples of such products are project plans, design documents, and code. The same authors define Software process maturity as the following:

> *Software process maturity is the extent to which a specific process is explicitly defined, managed, measured, controlled, and effective.*

Citation/definition 10: Software process maturity (Paulk et al., 1993)

One can distillate from this definition that increased maturity equals higher formality in documentation and operations. Therefore, maturity denotes the current growth in organizational capability and indicates the richness of an organization's software process, and the consistency with which the process is applied in projects (Paulk et al., 1993).

In addition to the CMMI characteristics, an immature software organization is characterized as reactionary, managers focusing on solving immediate crises, schedules and budgets not being accomplished, and compromised product functionality and quality in order to meet deadlines. In contrast, a mature software organization possesses an organization-wide ability for managing software development and maintenance processes (Paulk et al., 1993). Such organization can be distinguished by accurate communication between management and employees, planned and structured processes where all employees know their role, budgets and deadlines being met, calculated risk analysis, managers monitoring product quality and customer satisfaction, and decreased variability in predicted results. Based on this description, we notice that people are an important aspect to realize high software process maturity. However, unsuitable technology will only decrease the performance of those high performing teams. The organization's technology should rather be able to support these teams in their operations. Therefore, technology must be considered as well, in order to achieve software process maturity as an organization.

As an organization gains in software process maturity, it institutionalizes its software process via policies, standards, and organizational structures (the increasing of formality). *Institutionalization* entails building an infrastructure and a corporate culture that supports the methods, practices, and procedures of the business, so that they endure after those who originally defined them have gone (Paulk et al., 1993).

Higher software process maturity is achieved by following the defined paths of maturity models. Depending on the context and goal of an organization, different maturity models are relevant to apply. As organizations have the need for tangible knowledge on applying containers in their software delivery processes, the concept of an evolutionary path with different plateaus can be used. Such a concept results in a maturity model for software development processes, with the focus on gaining maturity by applying containers. In the prior section, we described SDLC. The results of that section are used to state maturity levels by the means of the CMMI structure. Additionally, we will determine on what position and to what extent containers should be used to gain maturity. Section 7 Design plan elaborates on how we are planning to realize such a maturity model.

3.11. Microservices architecture

Microservices architectural pattern is an approach for developing and managing a single application as a *suite of small services*, each running its own process and communicating with lightweight mechanisms (Lewis & Fowler, 2014). In practice, these lightweight mechanisms are APIs. The authors explain that the services are built around an organization's business capabilities, and are independently deployable by automated infrastructure. The microservices architecture (MSA) emerged from the need to update and run applications faster and more frequently than originally possible in conventional (application) architectures – in other words 'monoliths'.

3.11.1. Microservices architecture vs Monolith architecture

Applications in conventional, monolith architectures consisted of services that behave like *libraries*. This means that services are linked together into a program (process) and are called using in-memory calls (i.e. a single process application). Therefore, different services in monolith applications are all continuously linked to each other. As a result, when an organization updates one of the services' code, the complete application has to be rebuilt and deployed again. The same applies for scaling of monolith applications in the application landscape. For example, Application A (App1) exists of Service 1 (S1), 2 (S2) and 3 (S3). Because of a public event, user demands of S2 majorly increase. In the monolith way of working, the whole application needs to be replicated multiple times to meet required capacity. Despite the fact that solely S2 is requested a lot, whether S1 and S3 are not.

In a MSA, services behave like *components*. This means that services are decoupled from each other, and are therefore independently deployable and scalable. Hence, the above scenario of App1 is handled differently. As the services of App1 are decoupled from each other, their synergy of independent services form together the application. In that case, updating S1 only requires to rebuilt and deploy that specific service, rather than rebuilding and deploying the whole application again. In regards to scalability, only the highly demanded S2 gets more instances launched to increase its capacity. This is performed by the means of deploying S2 instances in clusters of containers onto one or more servers. Behavior of both architectural patterns in this scenario is visualized in Figure 36. Characteristics are described in the upper part, whereas in the lower part an example is given per characteristic.

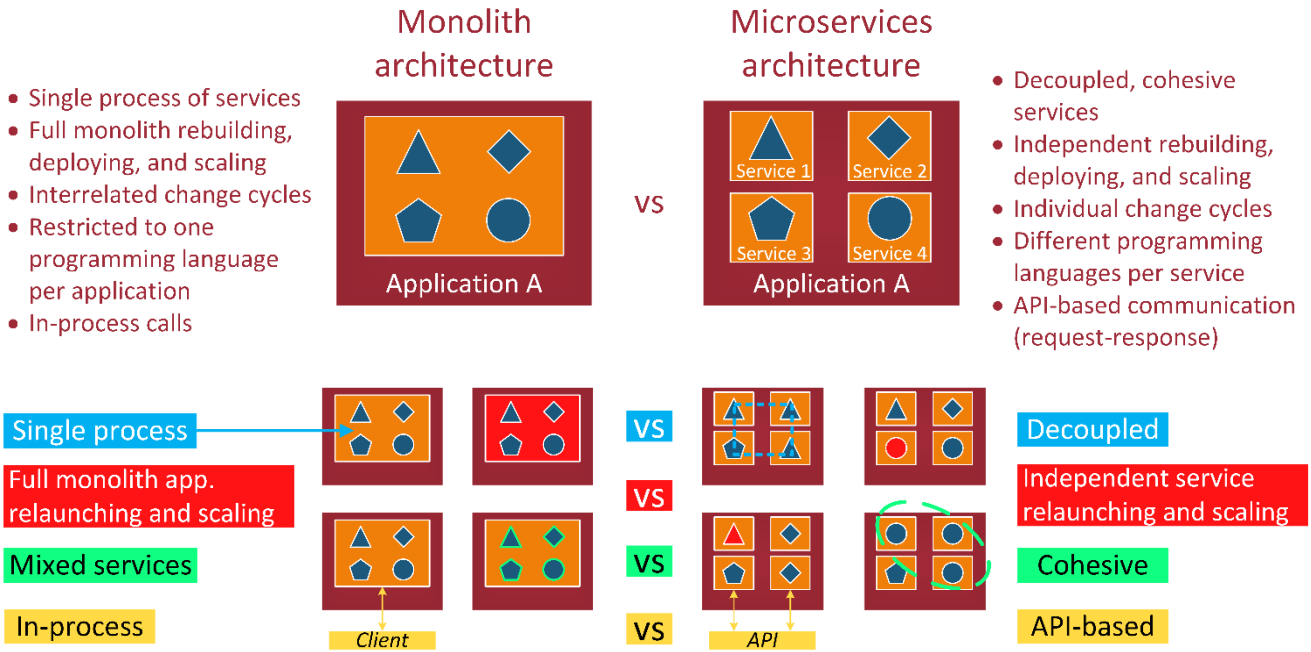


Figure 36: Monolith architecture vs MSA

Due to the decoupling of services, MSA brings different advantages. Cohesion means that a service's elements belong together. In other words, a service with higher cohesion is specialized to execute one or several tasks.

Low cohesion means that a single service contains too much different operations to execute, making the service slower and more prone to cause failures.

3.11.2. Decoupled service execution

As aforementioned, services in MSAs should behave as components. A component is defined as "*a unit of software that is independently replaceable and upgradeable*" (Lewis & Fowler, 2014). Subsequently, the same authors define services as "*out-of-process components who communicate with a mechanism such as a web service request, or remote procedure call define services*". With 'out-of-process', the authors mean that service dissociates itself from the monolithic single process architecture of a complete application. Subsequently, with the communication 'with a mechanism', the authors refer to the aforementioned remote APIs. These APIs exist in two types: 1) coarse-grained and 2) fine-grained, both having individual benefits and drawbacks (as denoted in Figure 37).

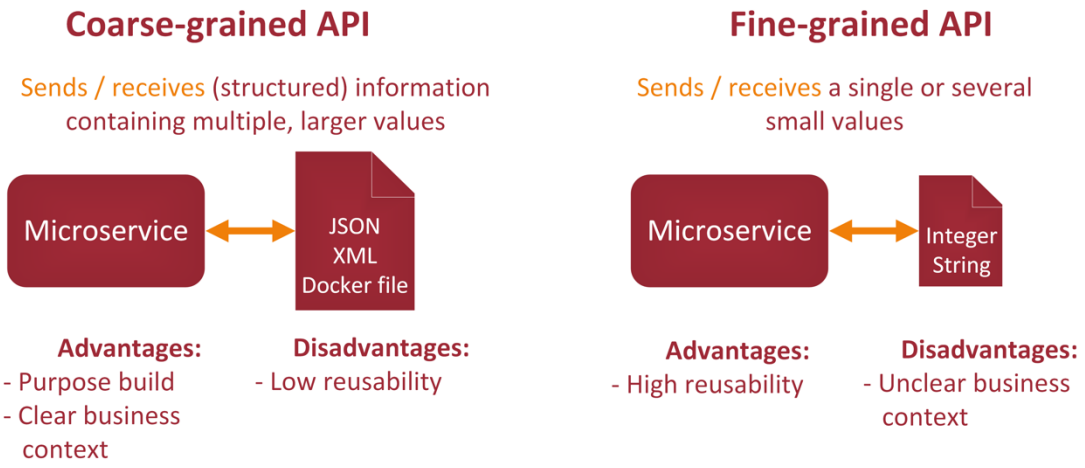


Figure 37: Coarse- & Fine-grained interfaces for service visualized

Figure 37 depicts the extent a service can be 'micro'. Services differ from handling files containing rows of values, to handling single values. The execution of a service is visualized in Figure 38. This figure shows that a service Receives input, modifies the input by Executing business logic (providing the service), and Sends the output to the designated receiver (RES).

Figure 38: Service execution - RES

Such services require an environment to be released in to run in production. Moreover, such services should be stateless (see section 2.7.3) in order to provide clean performance without the interference of other software, and to be able to scale-up/-down based to meet capacity demands. Besides that, the services in a MSA should be able to be independently rebuild, deployed, tested, and released into production again. All these requirements combined show that containers are the best concept to apply in the application landscape and infrastructure that enables a MSA to ultimately provide business value and increasing overall time-to-market.

3.11.3. The synergy of MSA, containers, and Infrastructure-as-code

As systems are decoupled from physical hardware due to the introduction of cloud, nowadays, routine provisioning and maintenance of infrastructure can be delegated to software systems (Morris, 2016). The arising developments from the cloud introduction enabled organizations to configure infrastructure more easily by using code. According to Morries (2016), this means that modern tooling can treat infrastructure as it is

software and data, instead of physical hardware. In other words, the introduction of Infrastructure-as-Code principle (IaC). Meandering with this principle is the concept of containers, as containers are configured, managed, launched and destroyed by code. Instead of configuring conventional infrastructure (e.g. VMs) through executing predefined tasks – both software- and hardware-based –, nowadays, simpler coding in infrastructure scripts and configuration files is becoming the standard.

Due to IaC, infrastructure management is now programmatically. The combination of programmable infrastructure and programmable virtual environments (containers), whereas the latter is launched in seconds without having many restrictions in systems, platforms, and programming languages, fully utilizes the benefits of MSAs. Subsequently, such fully utilized MSAs enhances the continuous SDLC process of an organization, parallel to additionally exploiting software development tools for deploying and managing infrastructure. Therefore, MSA and containers change the way of managing both an organization's SDLC process, as its underlying application landscape and infrastructure.

3.12. Enterprise architecture and Microservices architecture

Currently, enterprise architecture (EA) mainly focuses on conventional architecture styles regarding infrastructure and application landscape. Respectively, the Application and Technology layer. An example of such an older architectural style is the Service Oriented Architecture (SOA). However, the EA domain is not fully focusing on the newer architectural styles. Slowly, researcher start to explore the IaC and MSA principle for EA. As infrastructure (T) is now programmable, the Technology layer is now manageable through software tools. This is enabled by new school architecture styles. Especially with the combination of the cloud. The EA domain should incorporate these new styles, as the usage of the infra-as-code concept is ever increasing.

MSA is such a new school architectural style. Although, MSA originates as a software architecture style. Obviously, on the first impression, this is not the same as EA. However, when the MSA concept is scaled-up inside an organization's EA, the MSA concept is also applicable to both the deployment and (routine) maintenance of infrastructure and applications. Because of this capability to treat infrastructure as software, this architectural style (pattern) should be processed into the EA of an organization, in order for EA to advance as discipline. Especially through the combination of MSA, cloud, containers, and Infra-as-code. We believe this structure will be adopted by most (larger) organizations in the world, as cloud has become a standard to replace infrastructure, more applications are becoming cloud-based, and applications and services are getting decoupled as it increases agility of an organization. Examples of organizations exploiting this combination of concepts in their offered services are Amazon (AWS), Google (Kubernetes), Microsoft (Azure).

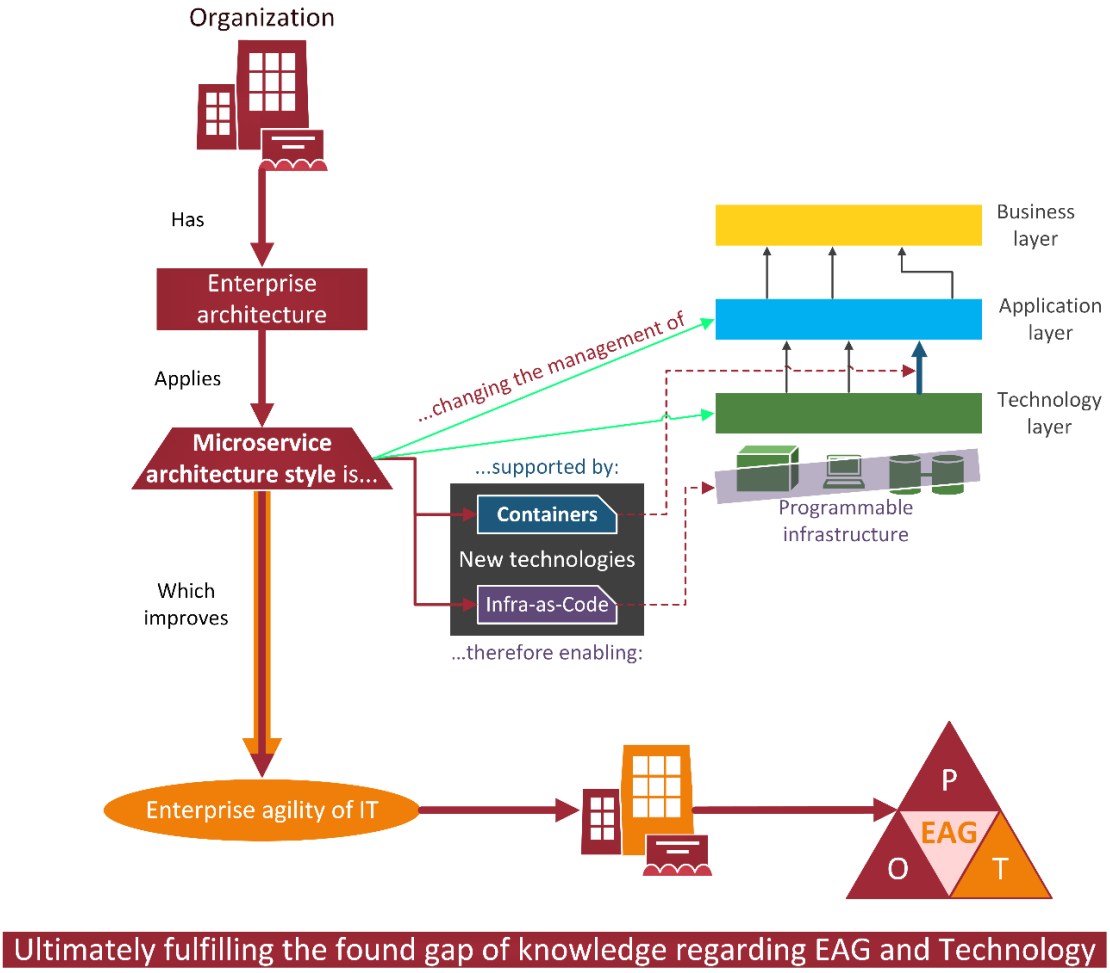


Figure 39: Synergy between EA, MSA, Containers and Infra-as-Code

Hence, as the managing of T of BAT changes, the discipline of EA is changing. Equivalent developments apply to the application landscape (A), intensifying the change to the EA discipline. Therefore, we state that applying microservices into an organization also changes the way an organization's EA is managed and executed. This should be considered by organizations at moment applying containerization is discussed.

4. Expert interviews

Section 4 contains two sub-sections. First, a brief version of the PI expert interview protocol is given in section 4.1. Secondly, results of the expert interviews are provided in section 4.2.

4.1. Expert interview protocol

The form of the used interview protocol is a semi-structured interview protocol. For this section, the expert interview protocol is abstracted into a high-over protocol showing the main topics that are discussed. This protocol is given in sub-section 4.1.1, where the main topics are accompanied with brief descriptions. We chose this structure to maintain clarity in the main document. The **full protocol** can be found in Appendix V.

4.1.1. General interview protocol

Introduction

Explaining about the master thesis project, and the goal and structure of this interview.

General

Consists of questions regarding the profile of the interviewee. In addition, his perspective on the role of enterprise agility is questioned, as to answer what agile aspects are.

Enterprise agility

Main findings of literature review are explained in this part. Interviewee is asked if he recognizes the statement about the lack of an explicit link between EAG and technology in his daily operations. If interviewee recognizes the statement, it is asked what his solution would be to define such link. If interviewee does not recognize the statement, it is asked what examples are of link between EAG and technology (e.g. agile technology examples).

Containerization

Characteristics and challenges of containers are discussed, and why this technology is currently popular. It is also discussed how containers can support EAG in their opinion, and what would be an example of a relationship between containers and EAG. Subsequently, interviewee's answer is linked to question about his perspective of an artefact regarding containers supporting organizations in order to enhance enterprise agility.

Finalizing

Finalizing questions are discussed, and follow-up actions are discussed.

4.1.2. Expert profiles

Four interviews have been conducted with experts from the field of practice. Three out of four interviewees are *Senior Managers* in the field of technology (ten plus years of experience in technology). The other interviewee is active in the same field and has the level of *Manager* (seven plus years of experience). Based on their level of expertise, the interviewees' opinion are perceived as valid for this research.

4.2. Expert interview results

Figure 40 depicts an overview of the summarized results of all four interviews. Results for each interview are denoted per row regarding the main discussed topics. Additionally, containers benefits according to the experts are given. The numbers in each benefit show which interviewee mentioned what benefit. The most important findings are the fact that all interviewees mentioned that the field of practice is in need of an artefact that provides support for implementing containers in the software delivery process. Reason for this need is the fact that most organizations are currently experimenting with applying containers in their operations.

The expert interview results are presented in a model. Most important topics are denoted in this model. Full elaborations of all interviews can be found in Appendix VI. Additionally, the interviewees were asked to denote the SDLC phases where containers could be applied. These results are drawn on a paper and shown in Appendix VII.

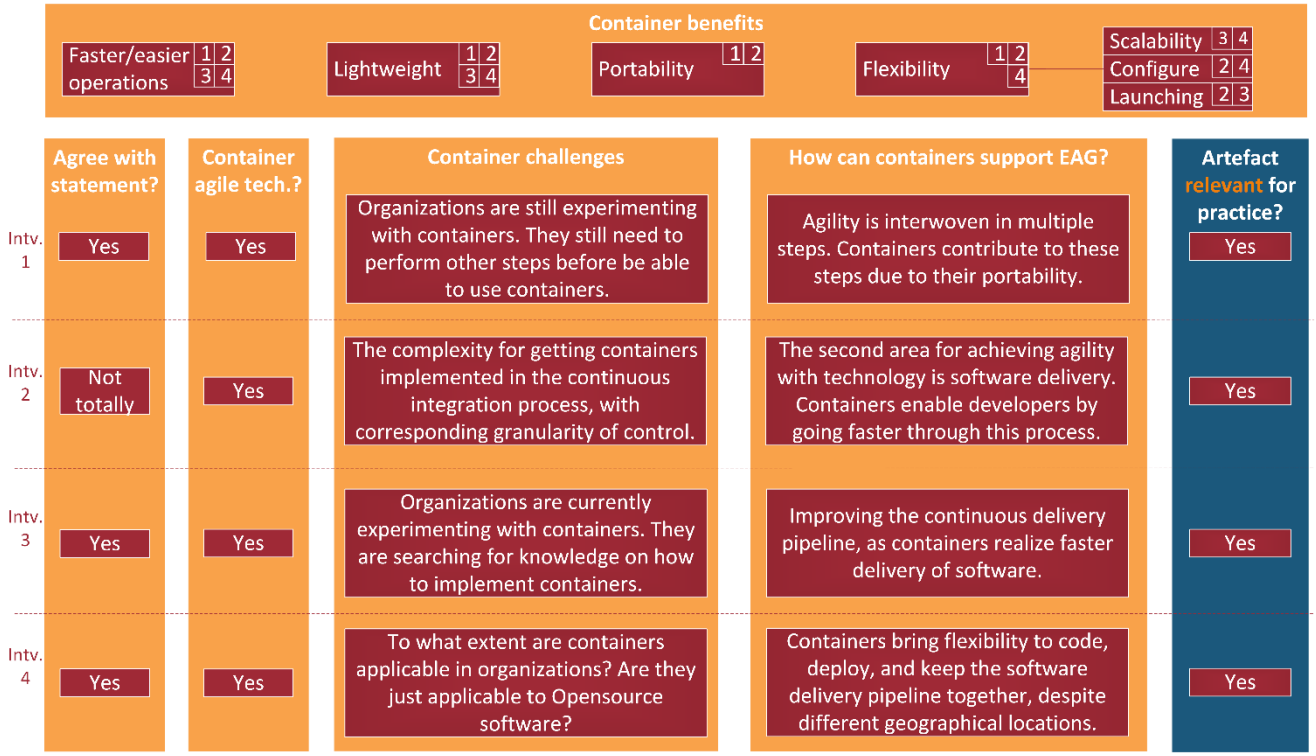


Figure 40: Overview of interview results

5. Conclusion of Problem Investigation

5.1. Findings literature review

5.1.1. Main concepts

The domain of agility is still subject to change. There are, however, main agility attributes and most important enterprise components defined with respect to EAG. These components are the aforementioned OPT concepts. As research regarding agility is dedicated to both O and P, we found no research dedicated to link technology (T) regarding agility. Only the importance of technology is indicated by several researchers.

Yusuf et al. (1999) states that in reference to agility, an organization should be aware of arising technologies, pursue leadership in technology use, and obtain skill and knowledge regarding technology. Besides that, Tseng and Lin (2011) created a conceptual model around agility components. Here, the authors stated that technological innovations and information integration are main agility components. Finally, Pal and Pantaleo (2005) argue that infrastructure must be adaptive. This shows that only the importance of being aware of upcoming technologies and being able to apply these are described, accompanied with one characteristics for infrastructure: *adaptivity*. We did not find further explanations around which technologies enable agility or contain agile aspects.

Nowadays, different technologies exist that contain agile aspects or that can support the agile WoW through their functionality. Examples are Application Programming Interfaces (APIs) and containers. In short, an API provides a set of code (protocols, tools) that enable simple interaction mechanisms between different applications. Hence, instead of developers building their own code to let applications communicate with each other, developers are provided with an API where they can conform their code to. This enables development to deliver software faster. Besides that, containers also enable organizations to become (more) enterprise agile. As aforementioned, containers are lightweight, launched in seconds instead of minutes as VMs require, use less hardware resources, share architectural components while maintaining spatial isolation, and support agile development processes. In other words, containers are realizing an agile virtual runtime environment.

However, we found that APIs and containers are not explicitly linked to a concept around ‘technology agility’ or ‘agile technology’ in the scientific field. Technology keeps being a (secondary) mean for other studies to improve the topic of interest. Studies only mention the importance of technology for organizations. This denotes the current gap of knowledge regarding an explicit link between agility and concrete examples of technology. Therefore, we hypothesize that establishing a link between containers and technology agility would add the ‘T’ of OPT which completes the scientific spectrum of agility. This scenario is depicted in Figure 41. Section 5.3 *Proposal in a nutshell* elaborates on such link in through the proposed artefact.

Figure 41: OPT components complete

Moreover, new knowledge for Technology can subsequently be applied for the field of EAG. The current scientific gap around technology regarding both agility and EAG is in that case partly solved. Considering the situation where EAG and technology is solved, solely then an organization can become fully enterprise agile from the scientific perspective.

5.1.2. Additional concepts

SDLC comprises the complete lifecycle of a piece of software. During its development, virtualization technologies can be applied onto two levels: 1) Process level: in the SDLC process, and 2) App & Infra level: to provision the SDLC process. Besides that, we found two types of activities that are included in the SDLC phases, which are Development and Operation activities. Development activities consist of all tasks and activities that are executed during the development of software. Operation activities comprise all tasks that are related to the management and monitoring of running applications in the production environment.

We examined multiple existing maturity models. Most characteristics of the models were similar to each other. Although, we also found several differences between the models. All models consisted of a structured, evolutionary path of growth that improves maturity of a process on incremental base. Each maturity level contained a set of process goals, or synonyms that indicated specific objectives per maturity level. Achieving a maturity level results in establishing new components which adds process capabilities, or improving process capabilities.

The overall path of growth, or evolutionary path share the same kind of start- and end positions. In general, first maturity levels indicate basic processes or functionalities, and no specific- or low component performance. In contrast, higher maturity levels indicate structured and formalized processes or operations, that additionally focus on continuously improving. Throughout the maturing process, all examined maturity models also share an equivalent structure when abstracted. Improvement from the low level basic configuration aim to enable standardization. Next steps are adding more details (improving capability), to create measurable indicators, which is followed by monitoring and further improving the process or operations performance and quality.

This shows that most of the maturity models' structure share high resemblance, whereas the largest differences are perceived as the variables put into the structure. We also found that most maturity models indicate higher formality in documentations and operations in high maturity levels. In addition, we conclude that achieving a maturity level results in either one of the following outcomes:

- **Functional addition**: complete new component is added;
- **Quality improvement**: existing component performance is improved.

Therefore, higher maturity denotes the richness of an organization's process or operation.

The microservices architectural pattern enables to manage the services of applications as individual pieces of software that can independently be deployed and scaled, as services behave like components and the communication is API-based. This aforementioned API technology facilitates communication between a service and the other elements of the architecture.

Based on the literature findings regarding functionality and characteristics of both containers and MSA, we state that containers are the preferred option to use in the MSA pattern. This increases an organization's speed and agility regarding its technology landscape. Besides that, another concept that further enables this speed and agility increase, is the IaC principle. By this principle, infrastructure configuring is now programmatically, equivalent to the configuring of containers. Combining this principle and technology, fully utilizes the benefits of MSAs. Subsequently, a MSA improves an organization's SDLC on both Process level and App & Infra level.

To implement the MSA pattern, organizations have to consider their EA. However, the EA domain does not heavily focus on the technology layer. Moreover, due to the increasing usage of different cloud solutions, the way of handling infrastructure to provision an organization is getting more fit for improvements. Especially with the IaC principle.

Examining these development, we believe that considering EA is crucial to start with making an organization container-ready. The next step is to focus on the MSA pattern, while incorporating the IaC principle to both suit the integration of containers, as enable an architecture that fully utilizes the advantages of containerization. Therefore, traditional organizations who want to (start) apply(ing) containers should investigate the latter in their own operations environment.

5.1.3. Requirements maturity model

Based on the results of the PI phase (including the expert interview results in next sub-section), we created a table with the most important characteristics of 1) Maturity models, 2) SDLC, and 3) EAG. We use these characteristics as a start for the design iterations in the TD phase. During the selection of the characteristics per concept, we considered how these concepts and their characteristics relate to containerization.

We believe that institutionalization is an important characteristic for maturity models. However, we did not incorporate this characteristic in our list of requirements for the proposed CMM. Containers are infra-agnostic, and the proposed CMM does not consider corporate culture or specific software an organization uses. The proposed metrics solely concern container-related tasks and functionalities in the SDLC process.

Table 14: List of main characteristics for proposed CMM

Concept	Characteristic	Description and rationale
Maturity models	Structured evolutionary path A growth process	We adopt the structure of a structured evolutionary growth path. Logically, this corresponds with a *maturity* model. Secondly, we intend to provide a structured way in improving container usage for organizations. In such a path, we want to follow the growth path of a black box functionality or basic functionality and low performance configuration at lower levels, towards more structured and formalized configurations that describe container-based configurations or landscapes.
	Differentiating components Establishing components	The evolutionary path brings new components that add new functionality to the container configuration, or related SDLC configuration. This denotes both the found process improvements on incremental base, and the different capability or maturity plateaus. In addition, this shows the *functional addition* of a component.
	Maturity levels Process capability	Shows the different level of maturity and improvements. This includes components (functionality), capabilities, objectives, and the characteristics of components. The latter shows the quality improvement of an already enabled component in a landscape. Besides that, higher maturity equals *higher formality* in documentation and *operations*.
	Maturity gaining Incremental improvements and increase in formality	Incorporates the evolutionary path of growth with differentiating components, capabilities, objectives etc. Achieving maturity levels gains incremental improvements that can be new functional or quality improvements. We believe higher maturity levels should consist of descriptions about aspects of the configuration. This increases the formality of a process or operation.
	Hierarchy CMMI model elements	The CMMI provides a hierarchy of its structure of elements, showing how maturity levels are spread throughout multiple parts of a maturity model. We intend to (partially) adopt this structure, as we think this structure show a balanced 'waterfall of granularity', and has proven itself over time.
SDLC	SDLC process level	The use case we focus on during the design of the CMM. Main container applications are activities around deploying and testing.
	Development phases	The main phases of the SDLC process (Code, Build, Test, Deploy, and Release). This includes activities around development or maintenance of applications.
	Monitoring or operations phases	Activities around managing and monitoring of running applications in the production environment.
	SDLC application & infrastructure level	All activities around the provisioning of the SDLC process. Other processes in an organization are also provisioned through this level, although to keep it simple, we maintain the name for SDLC provision to avoid possible complexity.
EAG (Sherehiy et al., 2007)	Flexibility	The ability to pursue different business strategies and tactics, to quickly change from one strategy, task, or job to another.
	Speed	The ability to complete requirements of all other agile characteristics in shortest possible time. The ability to learn, carry out tasks and operations and make changes in shortest possible time.
	Responsiveness	The ability to identify changes and opportunities and respond reactively or proactively to them.
	Integration and Low Complexity	Close and simple relations between the individual system components, easy and effortless flow of the materials, information and communication between the system components, organizational structures, people, and technology.
Containers	*Rationalized hardware deployment*	*Infra-agnostic application runtime environment*	*Configure through code*	*Stateless use case support*	*Lightweight OS emulation*	*Customizable images (libs, bins, runtime packages)*

5.2. Findings expert interviews

The experts from the field denoted that currently, organizations are in need of tangible knowledge regarding the application of containers. Organizations are in different stages in the use of IT. In other words, the IT maturity level of organizations differentiate. Based on this maturity level, organizations need specific knowledge in order to successfully apply containers in their software delivery process. Therefore, we asked the experts about a maturity assessment regarding the implementation of containers to ultimately improve enterprise agility. All experts indicated that such maturity assessment would be beneficial for organizations in the field of practice.

However, all experts indicated that such maturity assessment should be combined with improving continuous delivery and integration of the software delivery process. The experts stated that the software delivery process is the abstract process, containing different phases. These phases are code, build, deploy, test, and release. In these phases, multiple processes exist that have to be finished to complete a phase. Based on the aforementioned characteristics of containers in section 1.1 & 2.7, we hypothesize that containers can be applied in different manners in these software delivery processes to achieve (more) enterprise agility regarding technology. An elaboration of this hypothesis is visualized in section 5.3.

5.3. Proposal in a nutshell

Containers are a promising technology for organizations. However, besides the high-end IT leading organizations (e.g. Google, Spotify, Amazon), the experts mentioned that most traditional organizations do not have in-house knowledge about the latest technologies. This means that, as containers are a relative young IT concept, organizations do not know how to start with containers. Based on organizations' current use of VMs, and arrangement of their SDLC process, the configuration to start applying containers is different. In other words, dependable on the IT maturity of an organization, organizations should apply containers differently.

According to the results of the problem investigation, the field of practice is in need of tangible knowledge regarding the implementation of containers. Besides that, the scientific domain requires an explicit link between EAG and technology. We can combine these needs into a maturity model regarding the implementation of containers in SDLC. Such a container maturity model enables organizations to integrate containers into their infrastructure, and to subsequently support their EAG. Figure 42 visualizes this concept.

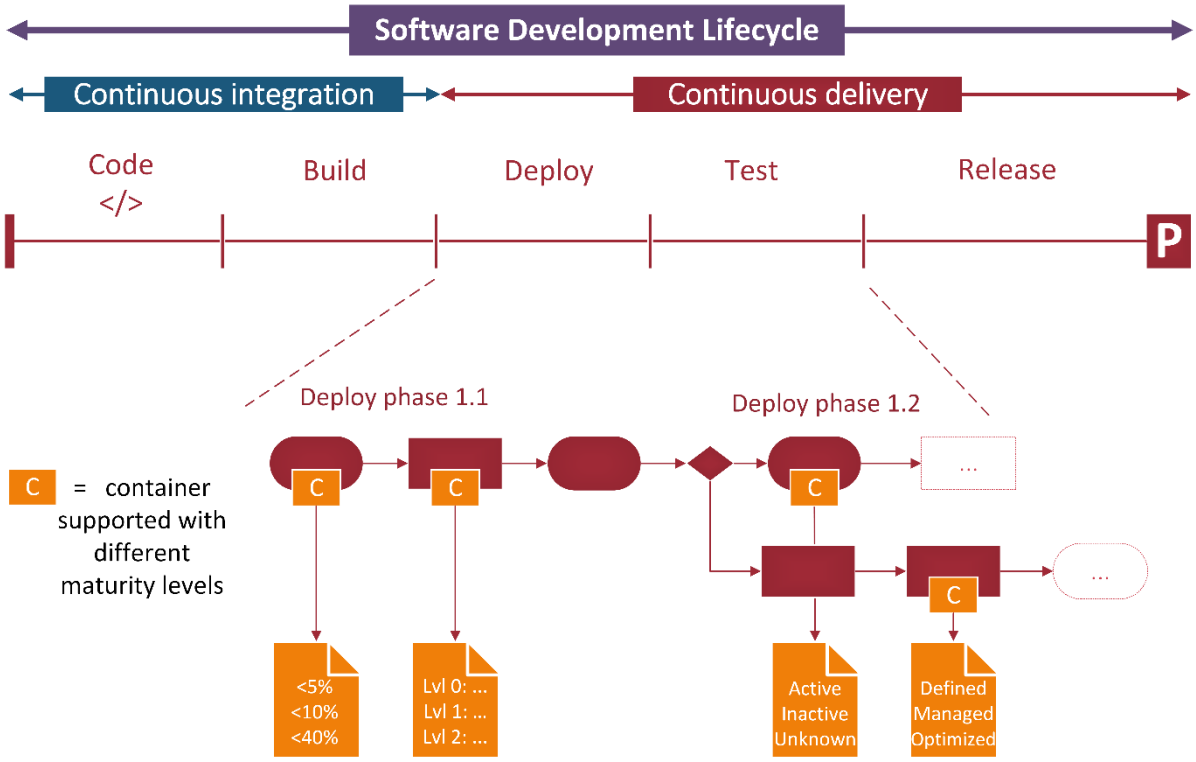


Figure 42: Initial design of proposed artefact: container implementation maturity model

We propose to further conduct research on SDLC, continuous delivery and integration, maturity models, and other related concepts. Subsequently, we can use the information resulting from this additional research to design a **Container Maturity model (CMM) focused on the SDLC process**, to ultimately enhance enterprise agility from a technology perspective. Next paragraphs briefly describe the proposal.

SDLC exist of processes from writing code, to release a product to end-users. In these processes, different stages exist where software needs to be built, tested, integrated etc. All these steps and stages are executed through different means, depending of the organization. Most organizations still use VMs to support the steps and stages. However, they can also apply containers in these steps. Moreover, this can range from basic configurations using some container-functionalities, to automated and integrated containerized (development) environments.

Using containers in an environment that suffice to container-specific requirements, results in organizations taking advantage of the containers' agile enabling benefits. By assessing the organizations' software development configuration and stating their maturity level, and subsequently providing the organizations

knowledge on how to start using containers based on their maturity level, organizations can increase their enterprise agility regarding technology. Hence, the proposed maturity model exists of three main components:

1) **Pre assessment**: determines the applicability of containers for an organization based on organizational characteristics (e.g. operating market, products, and the amount of usage of IT).
2) **Maturity assessment**: assessing the as-is configuration of an organization's SDLC and stating maturity levels for different areas of the SDLC.
3) **Container implementation guidelines**: recommending an improved, to-be SDLC configuration where containers are applied, based on the 'Results' (blue in Figure 43) of the maturity assessment.

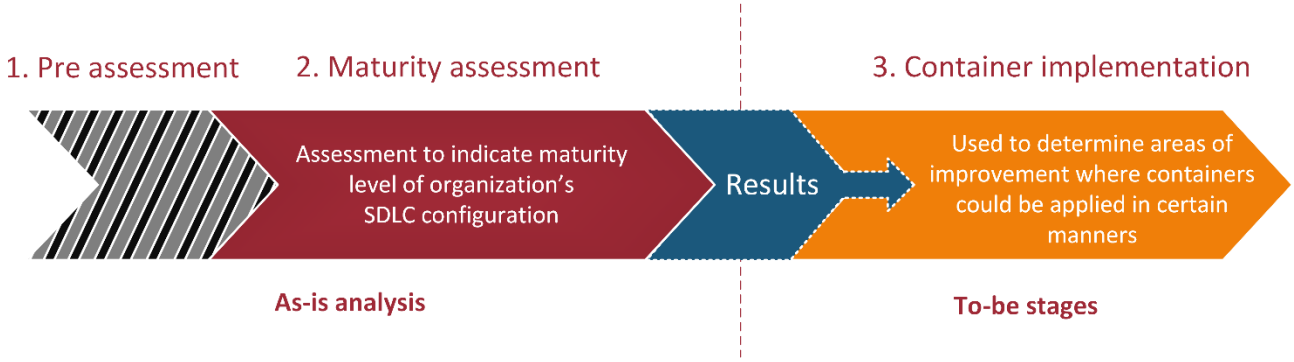


Figure 43: Components of the proposed maturity model

The overall SDLC consists of several development practices. Most known practices are Continuous Integration (CI) and Continuous Development (CD), which organizations are increasingly using as they embrace the agile WoW. Briefly, CI comprises developers integrating their work on daily base into a shared repository, and CD ensures that software is always production ready throughout its complete lifecycle. Considering the continuous 'stream' of (small) tasks that have to be performed to realize these practices, containers in particular provide improved support for these kinds of tasks, rather than virtual machines provide.

The proposed maturity model will complement the field of practice with an applicable product, and hence fulfil the found need of knowledge regarding the application of containers. In addition, continuing this research leads to a contribution to the scientific domain, as this maturity model realizes an explicit link between EAG and containers (technology) from a scientific perspective. This ultimately fulfills the found gap of knowledge regarding EAG and technology (T of the aforementioned OPT of section 3.4). Therefore, this means that organizations can become fully enterprise agile from the perspective of the scientific domain. Figure 44 depicts this situation through an abstract manner.

Figure 44: OPT fulfilled

Treatment Design

6. Introduction of Treatment Design

The second phase of Design science is the Treatment Design phase. As the author of Design Science describes, TD is the phase to design one or more artefacts that could be used to 'treat' the defined problem context (Wieringa, 2014). Consequently, the corresponding goal is to improve the situation of a group of stakeholders, which are in our research project the (traditional) organizations who want to apply or start applying containerization.

In this section, we briefly elaborate on how we executed the TD phase, the structure of the TD phase, and shortly highlight the contributions we realized by this research project. These contributions are:

1) Research around involving virtualization technologies in SDLC and an organization's EA;
2) Container Maturity Model (CMM), including the concepts of EAG, SDLC, and containers;
3) Process Deliverable Diagram of the CMM;
4) Validation of the CMM by experts with different specializations.

Contribution 1) is described in the literature review through the sections about the additional concepts and corresponding conclusion. The remaining contributions 2), 3) are described in this section. The final contribution 4) is described in section 12.

We combined TD with Treatment Validation into one single phase, as we executed expert interviews and validation interviews simultaneously. We further elaborate on the rationale for this decision in Section 7 Design plan and section 12 Validation of the CMM. To maintain clarity in the structure of the research and this document, we decided to split the TD and Treatment validation parts from each other in the document. However, we included the TV introduction into this introduction for the same reasons.

The structure of the TD phase (page 79-101) is as follows:

- **Section 7 Design plan** describes the design plan we created to design the artefact. At first, the method is defined, which includes descriptions about the hypotheses we want to test, and descriptions around the different design cycles (iterations) we stated. Secondly, the section continues with how we aim to analyze the gathered data. A taxonomy is presented and explained for analyzing the interviews.
- **Section 8 Data analysis** elaborates on the results we found during the iterations. For each iteration, we described the main results and provided visual representations about the state of the CMM during its design process. We facilitated more detailed elaborations of each interview in Appendix XII, XIII, XIV.
- **Section 9 Results: Container Maturity Model** is dedicated on describing the CMM. We included high-level descriptions about the CMM's overall structure and complete list of concepts, as well as detailed descriptions about the two assessments, different areas, and the defined metrics. In Appendix X and X, we described additional details about each metric and corresponding maturity levels.
- **Section 10 Results: Process Deliverable Diagram of CMM** is an additional contribution we realized besides the CMM. This section described how the CMM sequence works, what deliverables are realized in each phase and how these deliverables relate to each other. We believe that such a diagram provides useful indication for CMM users, and that it supports future developments of the CMM.
- **Section 11 Results: The CMM linked to Enterprise Agility** is the final section of TD. In order to link our final CMM v1.0 version to EAG, we intended to link all metrics of the CMM – which we grouped in topics (see section 11) – with the found EAG attributes of Sherehiy et al. (2007).
- **Section 12 Validation of the CMM** further elaborates on the process of validating our designed CMM. As mentioned before, we also explain our decision to combine TD and TV in the expert interviews.
- **Section 13 Reflection on the Research process** facilitates our reflection on the executed research process of all design science phases. We focused on the research process itself, as its research validity.

7. Design plan

7.1. Method

We defined the following hypotheses:

> **H_0**: *Integrating containers in the SDLC process does not increase support[12] for organizations' enterprise agility regarding business and IT.*
>
> **H_1**: *Integrating containers in the SDLC process does increase support for organizations' enterprise agility regarding business and IT.*

To test these hypotheses, we executed different types of expert interviews through multiple iterations. We intertwined the design process with validation sessions, as we found that we required additional knowledge on SDLC since the literature review alone was insufficient. Therefore, during the interviews we asked the experts about their knowledge on the concepts, and about their opinion on the presented CMM concept versions. Each iteration used a CMM version to present the experts, and resulted in a further developed CMM version.

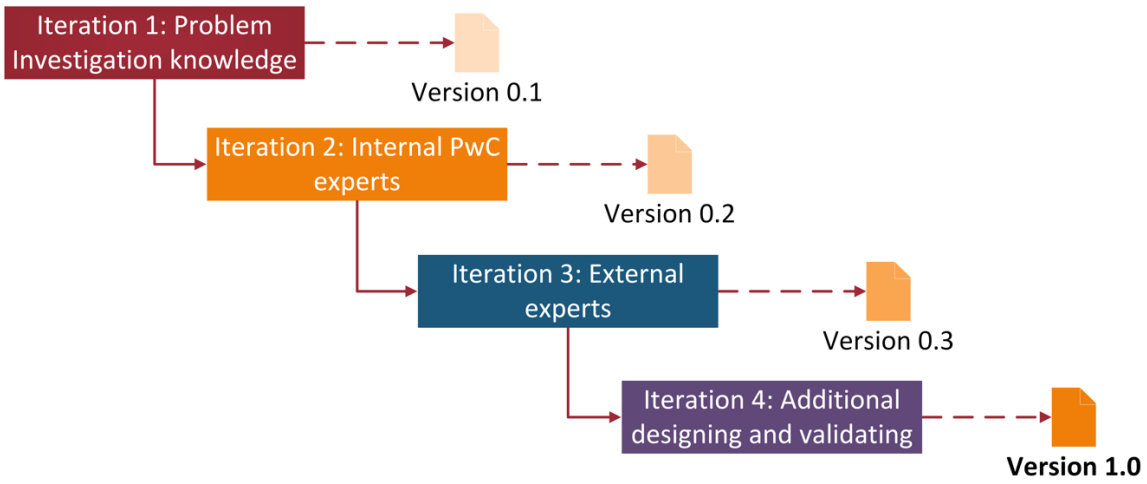


Figure 45: Iteration structure

Subsequently, we analyzed the interviews using NVivo 12 (see section 7.2 Interview analysis). During the iterations, we designed and evaluated the CMM, which was a process of incremental nature. After each iteration, a new, further refined version of the CMM resulted. We defined the following iterations for the Treatment Design phase:

- **Iteration 1**: Designing the CMM based on all research conducted during Problem Investigation phase.
- **Iteration 2**: Designing and validating the CMM based on feedback from internal PwC experts.
- **Iteration 3**: Designing, validating, and testing the CMM based on feedback from external experts.
- **Iteration 4**: Additional designing and validating the CMM based on feedback from both internal and external experts.

Figure 46 on page 83 visualizes the structure of all iterations, including their interdependencies to resulting in- and output.

7.1.1. Iteration 1: CMM based on Problem Investigation knowledge

The first version of the CMM is based on all knowledge (both literature and desk research) around containers and SDLC that we gathered in the PI phase (this includes the list of main characteristics). In addition, we held two unstructured brainstorm sessions with two internal PwC experts who vary in expertise. Results of these sessions we also processed into the first version of the CMM. We used the resulting version of the CMM (v0.01) as a base for the next iteration to ask interviewees about their opinion regarding the model.

[12] '*Support*' means supporting an environment that fosters enterprise agility, as containers enable a faster and more flexible SD environment.

7.1.2. Iteration 2: CMM according to internal PwC experts

As aforementioned, this iteration focuses on the perspective of internal PwC experts with different specializations. We used a semi-structured interview protocol – Protocol IT2, which can be found in Appendix XV – to discuss the CMM and find out what experts from the field think of the model. Subsequently, we used the experts' feedback to further modify and refine the model, in order to increase the model's validity. Hence, the CMM version resulting from this iteration – CMM v0.02 – is a further developed model. We used this new version to start with the third iteration.

7.1.3. Iteration 3: CMM according to external expert

Iteration 3 further elaborates on the opinion of experts from the field of practice. During this iteration, we performed an extended semi-structured interview of a duration of two hours with an expert from a large, international bank (Protocol IT3 can be found in Appendix XVI). The expert specializes in the infrastructure, application, and development domain. During this interview, we discussed each component and metric of the complete model. We questioned the expert's opinion and perspective regarding each part of the model. We use the expert's feedback to test our elaboration of the CMM, and to further refine the model according to the given feedback. Finally, the processed information resulted in CMM v0.03.

7.1.4. Iteration 4: Additional designing and validating

At first, we aimed to validate the model through validation sessions with experts in the fourth iteration. However, as further explained in section 12 and 13, we decided to perform additional expert interviews to extent the discussion about the model's content, and consequently gather more knowledge to further design the CMM. The outcome of this iteration results in CMM v1.0, which can be found in Appendix IX.

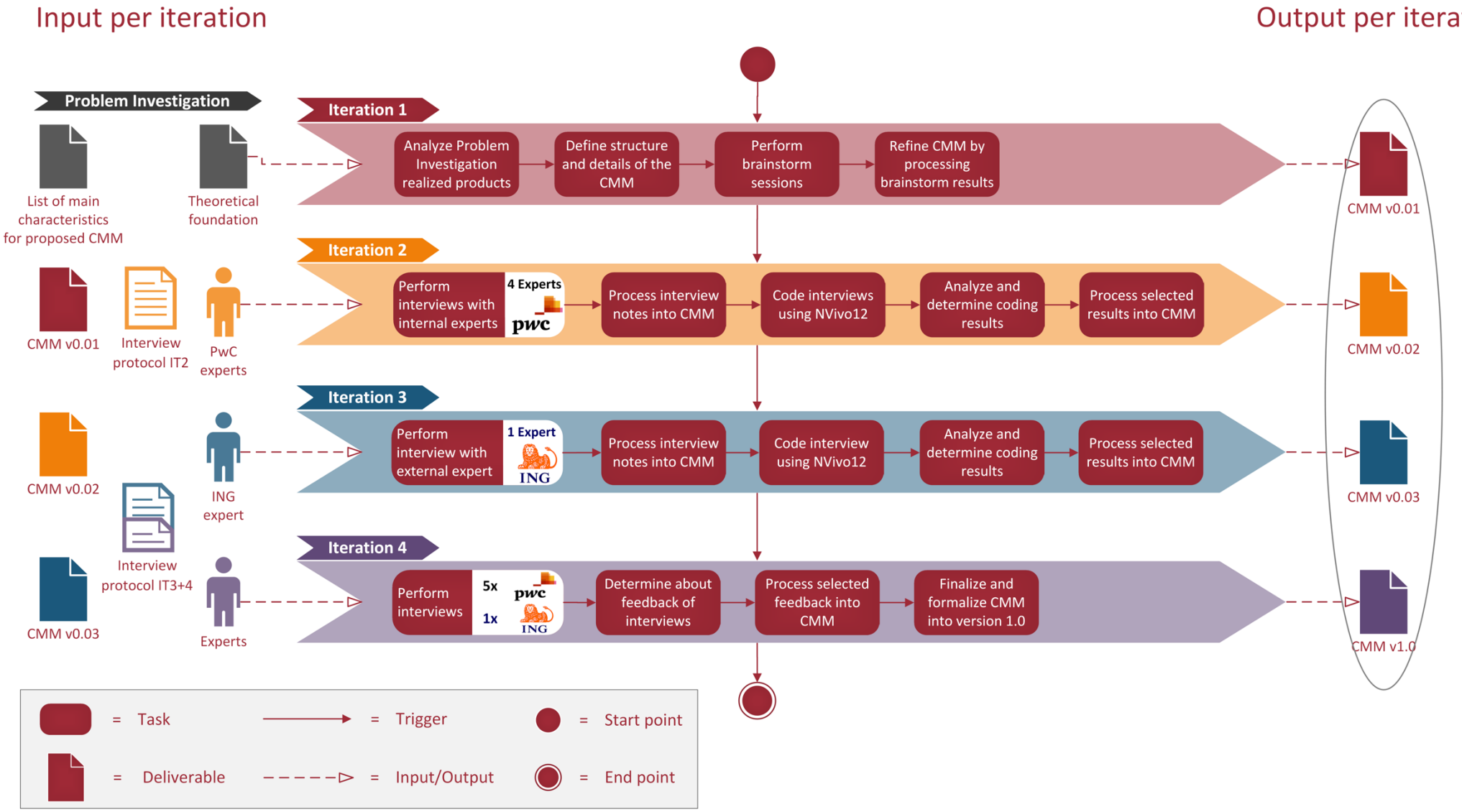


Figure 46: Treatment Design: Iterations and CMM deliverables

7.2. Interview analysis

All interviews of IT2 and IT3 are recorded[13]. We used NVivo 12 to analyze these recordings by tagging relevant and important parts of the interviews. A taxonomy of topics is created to implement in NVivo 12 to execute the analysis (section 7.2.2 & 7.2.3). We provided a brief explanation of NVivo 12 and its usage in section 7.2.1.

7.2.1. Analysis tool: NVivo 12

NVivo 12 is a tool that enables users to analyze transcriptions and audio files of performed interviews. A taxonomy can be created that is used to 'tag' sentences in transcriptions, or tag periods of time in audio files about statements of interviewees. The tags can be compared with other transcriptions and enables users to perform both qualitative and, in limited form, quantitative analysis of data.

The taxonomy that is used for the audio files of our interviews in Treatment Design is provided in section 7.2.2. Topics denote the context of discussion that we aimed for to focus on. Our focus is positive and negative comments of experts, next to their suggested changes. These are called as *Aspects*. These kind of aspects support the verification by experts of the CMM. Aspects are detailed metrics that are used for tagging periods of time in the audio files (see section 7.2.2 & 7.2.3). To increase clarity on what to expect, we provided an example of such a tag and resulting audio file diagram in Figure 47.

- *In interview **IT2Int#7**, between 00:22:35 – 00:25:03, the interviewee suggests that Component X should be added (yellow) to Development Area [DEV], as it does contribute to the model.*
- *In interview **IT2Int#7**, between 00:51:47 – 00:54:56, the interviewee commented positively (red) on metric Automated rollback in Operations Area [OPS].*

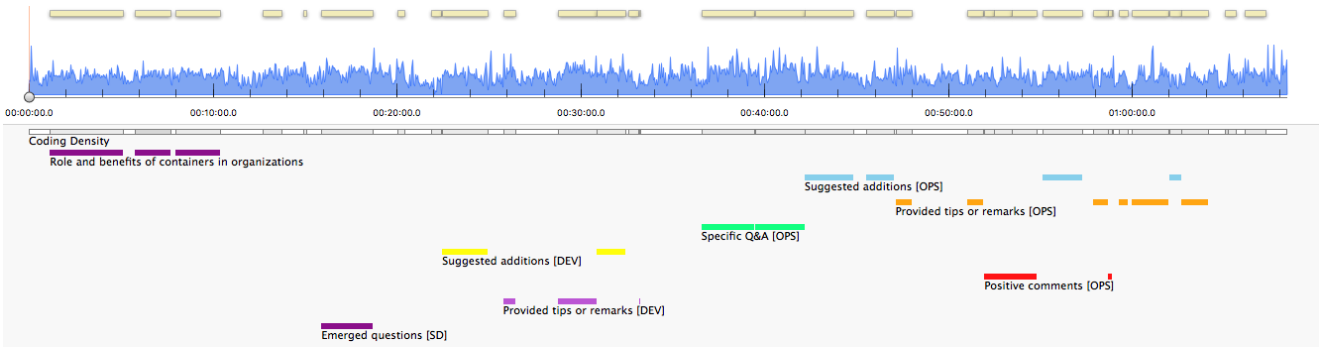


Figure 47: Example of visualization of tagged audio file

As depicted in Figure 47, the sound bars of all interviews including tags are given in Appendix XII, XII, and XIV. By doing this, we created visuals of each performed interview whereas one can easily distinguish what aspects are discussed during the interviews. Additionally, important statements of experts are quoted to strengthen results.

[13] During Iteration 2, IT2Int#3 got lost (100%) and IT2Int#4 got partial lost (~50%) due to technical issues. Hence, we switched from voice recorder solution for the remaining interviews.

7.2.2. Protocol: Taxonomy structure

The taxonomy is created using different perspectives. We defined two main topics of interest: CMM Area specific statements, and container trends statements. This is shown as the following:

❖ = **Topic**	• = **Sub-topic**	- = Area	○ = Aspect

❖ **Container trend statements**

- **Role of containers in organizations;**
- **Specific Q&A;**
- **Containers in SDLC context.**

❖ **CMM area statements**

- **Pre assessment**
  - General Area
    - Positive comments;
    - Suggested additions;
    - Negative comments;
    - Suggested removes;
    - Emerged questions;
    - Suggested changes;
    - Specific Q&A;
    - Provided tips and remarks.
  - Software Development Area
    - Positive comments;
    - Suggested additions;
    - Negative comments;
    - Suggested removes;
    - Emerged questions;
    - Suggested changes;
    - Specific Q&A;
    - Provided tips and remarks.
  - Application & Infrastructure Area
    - Positive comments;
    - Suggested additions;
    - Negative comments;
    - Suggested removes;
    - Emerged questions;
    - Suggested changes;
    - Specific Q&A;
    - Provided tips and remarks.
- **Maturity assessment**
  - Development Area
    - Positive comments;
    - Suggested additions;
    - Negative comments;
    - Suggested removes;
    - Emerged questions;
    - Suggested changes;
    - Specific Q&A;
    - Provided tips and remarks.
  - Operations Area
    - Positive comments;
    - Suggested additions;
    - Negative comments;
    - Suggested removes;
    - Emerged questions;
    - Suggested changes;
    - Specific Q&A;
    - Provided tips and remarks.
  - Output (BC) Area
    - Positive comments;
    - Suggested additions;
    - Negative comments;
    - Suggested removes;
    - Emerged questions;
    - Suggested changes;
    - Specific Q&A;
    - Provided tips and remarks.

7.2.3. Protocol: Taxonomy description

7.2.3.1. Taxonomy topic: Container trends statements

The first topic briefly elaborates on the role of containers in organizations and in the SDLC context. We also asked the experts about their opinion regarding the benefits of containers in specific use cases. All experts' answers combined contribute to the overall perspective we are creating regarding the increasing role of containers in software development and infrastructure. Table 15 briefly describes each aspect we focus on.

Table 15: Container trend statements taxonomy description

Aspect	Description
Role of containers in organizations	Moments where the expert denotes the role of containers in organizations.
Specific Q&A	Moments where the student asks an additional question, inspired by (new) information given by the expert about a specific metric or aspect of the CMM, or related topic.
Containers in SDLC context	Moments where experts state a suggestion to change a specific metric or aspect of a CMM area.

7.2.3.2. Taxonomy topic: CMM area statements

This topic also exists of two sub-topics, both having their own detailed components. The **1) Pre assessment** topic focuses on the questionnaire that rapidly indicates the profile of an organization, divided in three areas. This assessment should be able to be finished within five minutes by one senior employee or software development expert of an organization. The Pre assessment's and Maturity assessment's taxonomies are identical, as we were aiming for the same kind of feedback for both.

We discuss **2) Maturity assessment** for a longer amount of time during the interviews, as this part of CMM is more extensive. Equivalent to the Pre assessment, this assessment also exists of three areas, and share the same taxonomy aspects. The goal of the Maturity assessment is to indicate how an organization is using containers in their SDLC process, both production as non-production environments.

Table 16: CMM area statements taxonomy description

Aspect	Description
Positive comments	Moments where experts state a positive comment about a specific metric or aspect of a CMM area.
Suggested additions	Moments where experts state a suggestion to add a specific metric or aspect of a CMM area.
Negative comments	Moments where experts state a negative comment about a specific metric or aspect of a CMM area.
Suggested removes	Moments where experts state a suggestion to remove a specific metric or aspect of a CMM area.
Emerged questions	Moments where experts ask a new question about a specific metric or aspect of a CMM area or related topic. Asking for (further) clarity about a topic is not meant with this aspect.
Suggested changes	Moments where experts state a suggestion to modify a specific, existing metric or -aspect of a CMM area.
Specific Q&A	Moments where the student asks an additional question, inspired by (new) information given by the expert about a specific metric or aspect of the CMM, or related topic.
Provided tips and remarks	Moments where experts start to further elaborate about a topic, in order to emphasize or explain his answer, but also to elaborate on other related topics what results in additional useful information or improved understanding of the student towards a topic.

The Maturity assessment taxonomy aims for several goals: 1) the assessment questions about container usage in SDLC phases (Development Area), 2) it indicates the EAG regarding business and IT of an organization by focusing on CI and CD aspects (Operations Area), and finally 3) a business case with potential savings is presented (Output Area), based on the assessment's result.

8. Data analysis

8.1. Design iterations

In this section, we describe our findings of regarding all iterations. The fourth iteration is additionally described in section 12, as this iteration is focused on the Treatment Validation phase. Each sub-section comprises an introduction of the iteration, and a summary of main findings. In Appendix XII, XII, and XIV, we provided elaborations of each specific expert interview. These elaborations consist of summarized findings and outcomes of all tagged soundbars from the NVivo analysis.

8.1.1. Iteration 1: CMM based on Problem Investigation knowledge

The first iteration (IT1) was used to gather knowledge from personal executed research (literature findings, papers, performed semi-structured interviews, web sources), and PwC internal sources. Using this information, we designed the first version of the CMM (v0.1). We used this version as the base model to check with internal PwC experts regarding virtualization in iteration 2.

8.1.1.1. Processed results of IT1

We performed two additional brainstorm sessions with PwC experts. Through these sessions, we came to the first structure of the model. This structure comprises of different 'areas' in the Maturity assessment and 30 questions in the Pre assessment. Subsequently, we complemented the areas in detail using all prior and additional research of the Long Proposal. This resulted in CMM v0.1 of the first iteration.

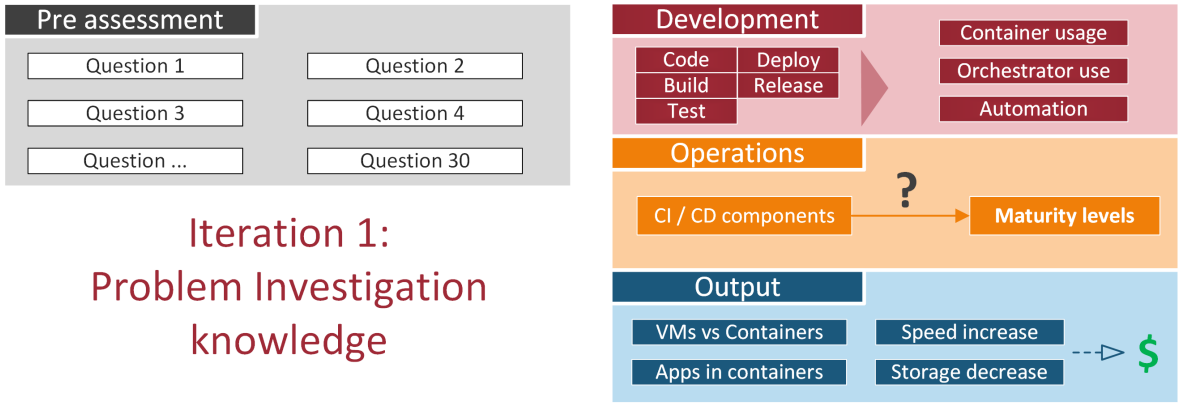


Figure 48: abstract visualization of results CMM design IT1

Table 17: Results

#	Summarized description of main findings	Rationale
1	We defined the structure of the CMM's first concept version during the brainstorm sessions internal PwC experts.	For the MA, we defined three areas (DEV, OPS, and OUT) that contained multiple metrics. The metrics are described throughout five maturity levels in metric specifications.
2	Another result of the brainstorm session was the addition of the Pre assessment to determine the container-readiness of an organization.	We added the PA as the first part of the CMM to serve as a quick pre-scan of an organization to decide whether that organization is benefited by continuing the CMM. We created PA questions based on our prior PI research.
3	We stated the content of the DEV and OPS area with information from the brainstorm sessions and prior PI phase research. Output was defined based on the brainstorm sessions.	In DEV, we aimed to focus on general container usage throughout the SDLC phases (the *what* of containers). This included metrics around processes and orchestrators. OPS would be focused on CICD components (the *how* of container usage).

During this iteration, we wanted to wait for the expert feedback of IT2, before deciding what maturity level structure to implement through the model.

8.1.2. Iteration 2: CMM according to internal PwC experts

The second iteration (IT2) is used to test the CMM with internal experts from PwC. Through a (semi-)structured protocol, we discussed all the facets of the CMM, where the experts gave their perspective and feedback. We aimed for new knowledge and insights, and verification on all details of the maturity model. With this iteration, we realized v0.02 of the CMM.

We used the taxonomy in section 7.2 as a support for executing the interviews. During the first interview (IT2Int#1), we received a large amount of relevant and informative explanations around containers and SDLC. We were able to use this information to increase our understanding of containers, SDLC, and the combination of both in such maturity model. Therefore, we decided to extend the taxonomy for the remaining interviews of IT2 with several aspects to increase the possibilities of our analysis. The following two aspects we applied immediately to the taxonomy during the analysis of the first interview:

- '*Specific Q&A*': as multiple questions which were asked by the interviewer emerged;
- '*Provided tips and remarks*': when asked about a component of the CMM, the interviewee started to explain that topic to emphasize or explain his answer, but also to elaborate on other related topics which resulted in additional useful information.

After the analysis of the first interview, we found that 'Suggested additions' and 'Suggested removes' were not sufficient in certain situations or for specific statements. For instance, when the interviewee suggested changing a component to another form, we perceived this as a new addition to, nor a removal of that component regarding the CMM. Hence, we determined to add ○ '*Suggested changes*' to the taxonomy.

8.1.2.1. Processed results of IT2

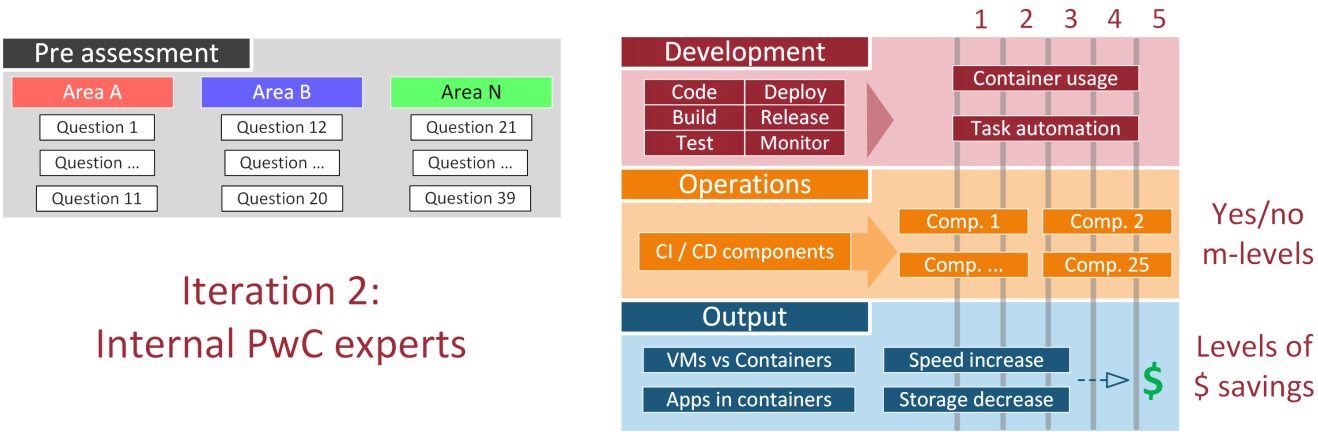


Figure 49: Abstract visualization of results CMM design IT2

Table 18: Results

#	Summarized description of main findings	Rationale
1	All experts argued that the PA should be shorter. A couple experts emphasized that crucial questions only should be presented.	We grouped the PA questions into three areas: 1) General – information about the organization and its sector, 2) Software development, and 3) Infrastructure and applications.
2	Several experts denoted that monitoring should be included to the CMM. More specifically, as final phase in the DEV area.	We added monitoring to DEV and hence provided this phase with same metrics as the other SDLC phases.
3	In the interviews, we found multiple times that (automated) testing should be split and further specified in the MA.	We split the testing metric into Application-dependent tests and Application-independent tests.
4	Based on the interviews, we were able to remove irrelevant metrics and complement the model with relevant metrics.	We removed several metrics from both the DEV and OPS areas. In addition, we further refined the maturity levels of seven OPS metrics.

5	We determined the maturity level structure for the MA.	The maturity model structure of Dreyfus (Dreyfus & Dreyfus, 1980), which include: 1) Novice, 2) Advanced beginner, 3) Competent, 4) Proficient, and 5) Expert.

8.1.3. Iteration 3: CMM according to external expert

During the third iteration (IT3), we discussed V0.02 with external experts. During the interviews, we immediately modified the model based on the feedback if possible. We had one interview of two hours scheduled at ING with an expert on the domains of application and infrastructure.

8.1.3.1. Processed results of IT3

During the design in the first two iterations, the Pre assessment contained three main areas including more than 30 questions. Based on the feedback from IT2 and IT3, we decided to reduce the amount of questions to the minimal required amount. This led to 11 questions in the Pre assessment. The three areas have been contained to denote of topic per question, instead of fulfilling a leading role.

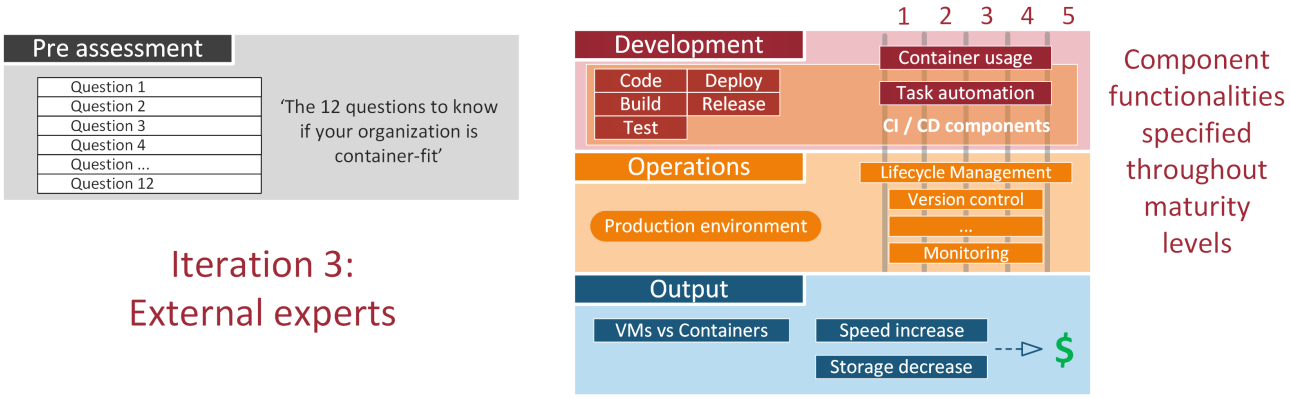


Figure 50: Abstract visualization of results CMM design IT3

Table 19: Results

#	Summarized description of main findings	Rationale
1	The expert denoted that we have to process the CICD components from the OPS phase towards the DEV phase. We also found more details on these metrics, which enabled us to further refine and remove currently included metrics.	We processed the CICD components in the DEV phase. The concerning components are now placed in the relevant SDLC phase, whereas functional behavior and system characteristics per components is described in the maturity levels.
2	OPS should include metrics regarding the live or production environment of applications that run on containers. These kind of metrics are around the concept of monitoring and lifecycle management.	We analyzed the recording of the interview and additional desk research and complemented the OPS area with new metrics regarding operations of running applications.
3	We discussed automation levels and stated a level structure.	We included this level structure in the MA, and modified each level structure to the concerning SDLC phase.

8.1.4. Iteration 4: Additional designing and validating

The additional expert interviews of this iteration were not recorder as in the previous iterations. We made this decision, since analyzing the recordings through NVivo 12 required an amount of time per interview we did not possess during this last iteration. Therefore, we made notes during the interview and processed most of the feedback immediately into the model. In order to track our gathered knowledge, we created for each additional interview an individual Excel version of the CMM and denoted all modifications with the color red. By doing this, we were able to track all the suggested changes to the model. Due to the restricted amount of time, we were forced to keep the report on the results brief for each interview.

8.1.4.1. Processed results of IT4

We visualized the main findings of the additional interviews (IT4) in an abstracted representation.

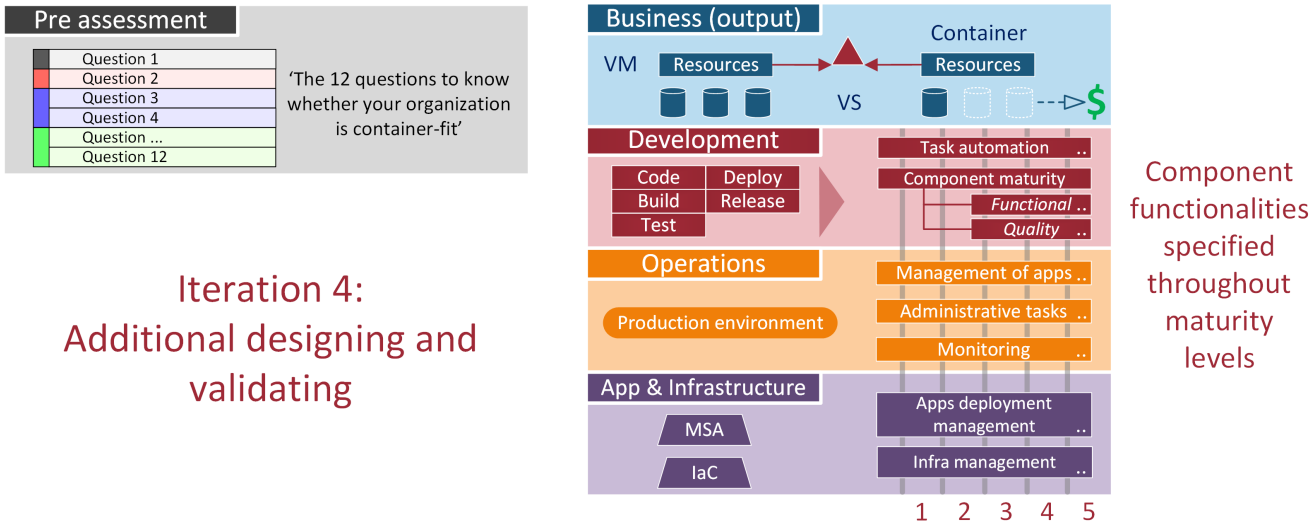


Figure 51: Abstract visualization of results CMM design IT4

Table 20: Results

#	Summarized description of main findings	Rationale
1	We received more feedback to further refine the PA.	We refined PA questions around virtualization, stateless applications, and hypervisor usage.
2	Several experts denoted that they missed software delivery in the CMM. The presented CMM did not correspond with the application of containers in the business domain.	Therefore, we added another area to the MA: Application & infrastructure (AppInfr). This area consists of metrics that are related to deployment management of applications, and the management of infrastructure elements.
3	Regarding the MA, more feedback was received regarding the DEV area metrics. With this feedback we further improved the metric specifications.	Especially level 5 metric specifications have been improved. In combination with the connection the AppInfr area, we further complemented remaining metrics.
4	The OPS area also received further attention, especially the higher maturity levels.	Equivalent to row 3, we further improved metric specifications of OPS.

9. Results: Container Maturity Model

Section 9.1 elaborates on the designed CMM. We provided an overall description, descriptions about both assessments including all the different Areas and their interdependencies, and we provided several disclaimers that are important to understand while applying the model in practice.

9.1. CMM main structure

Figure 52 depicts an abstract visualization of the main structure and sequence of the CMM. As shown in Figure 52, the CMM consists of the following three parts:

- **Pre assessment** (PA) – determines whether an organization is container-fit;
- **Maturity assessment** (MA) – the maturity assessment that assess an organization's SDLC configuration while being focused on containers;
- **Results matrix**[14] – depicts the results of the MA, including the maturity levels and business output.

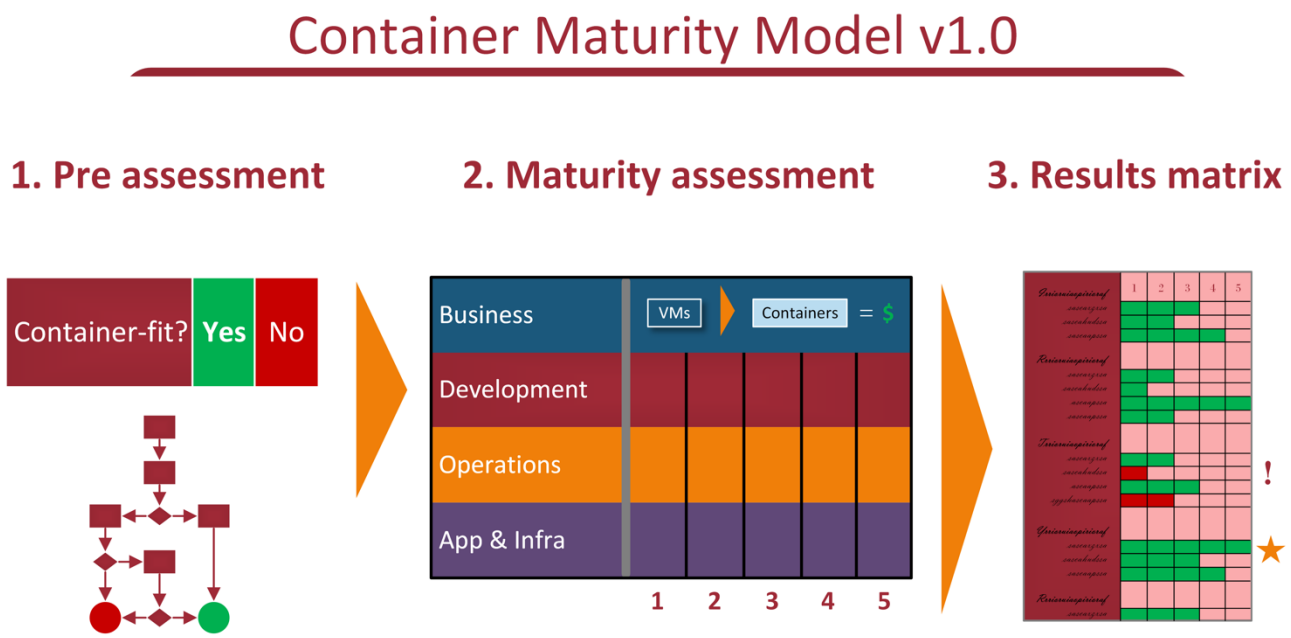


Figure 52: CMM complete: abstract visualization

Throughout the CMM, different concepts are used to denote content of the CMM. Table 21 on the next page contains all concepts that are used throughout the CMM. After that on the same page, in order to give a better understanding of the context of CMM and SDLC, we provided a conceptual model (Figure 53) that shows how these concepts are linked to each other.

[14] We envision to provide organizations applying the CMM, with a customized roadmap based on the PA's and MA's results, that indicates an optimal order of improving their SDLC configuration. To create such a roadmap with valid results, we were required to conduct additional research, which was not possible to fit in the remaining available time. Therefore, this aspect is denoted in section 15 Discussion and future work.

Table 21: Concepts used in CMM

Concept	Description
Pre assessment	A short assessment (approx. 5 min) that determines whether an organization is '**container-ready**'. Container-readiness means the extent an organization is able to benefit from container technology.
Maturity assessment	The main assessment that focuses on indicating the maturity of the SDLC pipeline, and operations that are performed while applications are running. This is measured through assessing CICD components in both SDLC and operations. Results are shown in the Results matrix.
Results matrix	A matrix that shows the achieved maturity levels for each Metric of an organization. By visualizing the results,
Maturity level(s)	The levels of maturity that can be achieved by an organization. A five-stage model, based on Dreyfus and Dreyfus (1980).
[name] + **Area**	A section in the CMM that focuses on a specific subject to capture or realize information about the assessed organization.
Topic	A group of metrics that together can be formed to a group.
Metric	An entity that is measurable to a certain extent and spreadable across five maturity levels. The metric supports the maturity indication SDLC or operations of running applications of an organization.
Component	The type of a concept that we processed into a metric. For instance, 'Artefact repository' is a metric in our CMM. However, when we refer to such concept without meaning the metric context, we use artefact repository as a 'component'.
SDLC phase	The incorporated phases of SDLC (Code, Build, Deploy, Test, and Release).
Business case	The business case that is generated by the results of Output Area. The main focus is comparing VM costs versus container costs, in order to estimate potential savings.

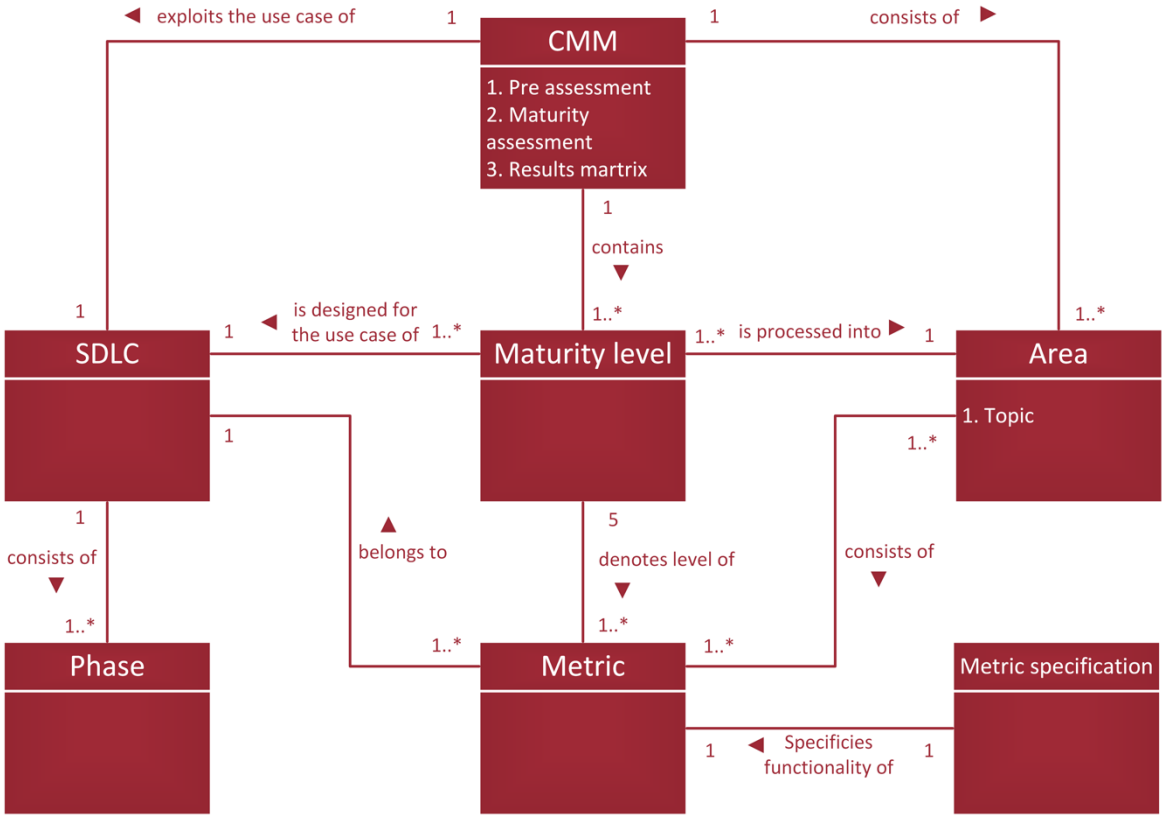


Figure 53: Conceptual model of CMM and SDLC

9.2. Pre-assessment

The Pre assessment exists of 11 questions that test an organization's container-readiness. Three areas are shown to denote the topic per question. We stated the questions in the following Table 22.

Table 22: Pre assessment Areas and questions

Pre. Area	#	Question	Answer
General	1	Is your organization developing or maintaining software?	y/n
	2	Does your organization also outsource parts of software development?	y/n
	3	How many employees are working in your organization?	no.
Software development	4	How many developers are working in your organization (excluding outsourcing)?	no.
	5	Does your organization make use of standardized deployment pipeline for more than 50% of their SD processes?	y/n
	6	How many times does your organization deploy each week?	no.
Application & Infrastructure	7	What type of cloud is your organization using? (Public, private, hybrid, on premise)	type
	8	What is the percentage of applications that are running stateless?	%
	9	Do you have the ambition to (partially) migrate to a microservices architecture style in the upcoming three years?	y/n
	10	What is the percentage of your applications' Functional Application Architecture that are container ready?	%
	11	What is the percentage of your applications that run by the means of virtualization?	%

9.2.1. General Area

The General Area [GE] is aimed at gathering information around the organization's employees and any outsourcing activities. The first question is the most important one, as when an organization does not develop software

9.2.2. Software Development Area

Questions of the Software Development [SD] Area indicate the importance and intensity of SD in an organization. Using this information, we can discover to what extent an organization can benefit from container technology in their SDLC process.

9.2.3. Application & Infrastructure Area

The final area [AppInf] of the Pre assessment comprises of questions regarding (running) applications and infrastructure of an organization. Through this, we can explore the current state of advancements of an organization towards container-ready environments. Especially, the balance of stateless and stateful applications and their corresponding architecture majorly support the indication of container-fitness.

9.3. Maturity assessment

The Maturity assessment exists of four areas, each having its own focus. Provided below are these areas, accompanied with an explanation:

1) **Business (Output)** [OUT]: results and potential savings by switching to containers (i.e. business case).
2) **Development** [DEV]: contains main phases of SDLC, including phase specific CICD components.
3) **Operations** [OPS]: exists of all tasks that are performed when containers are released in production.
4) **Application landscape & infrastructure** [AppInfr]: the hard- and software components that facilitate support for the SDLC process.

We denote with 'pipeline' the process of SDLC, whereas either software is developed or delivered to the requester.

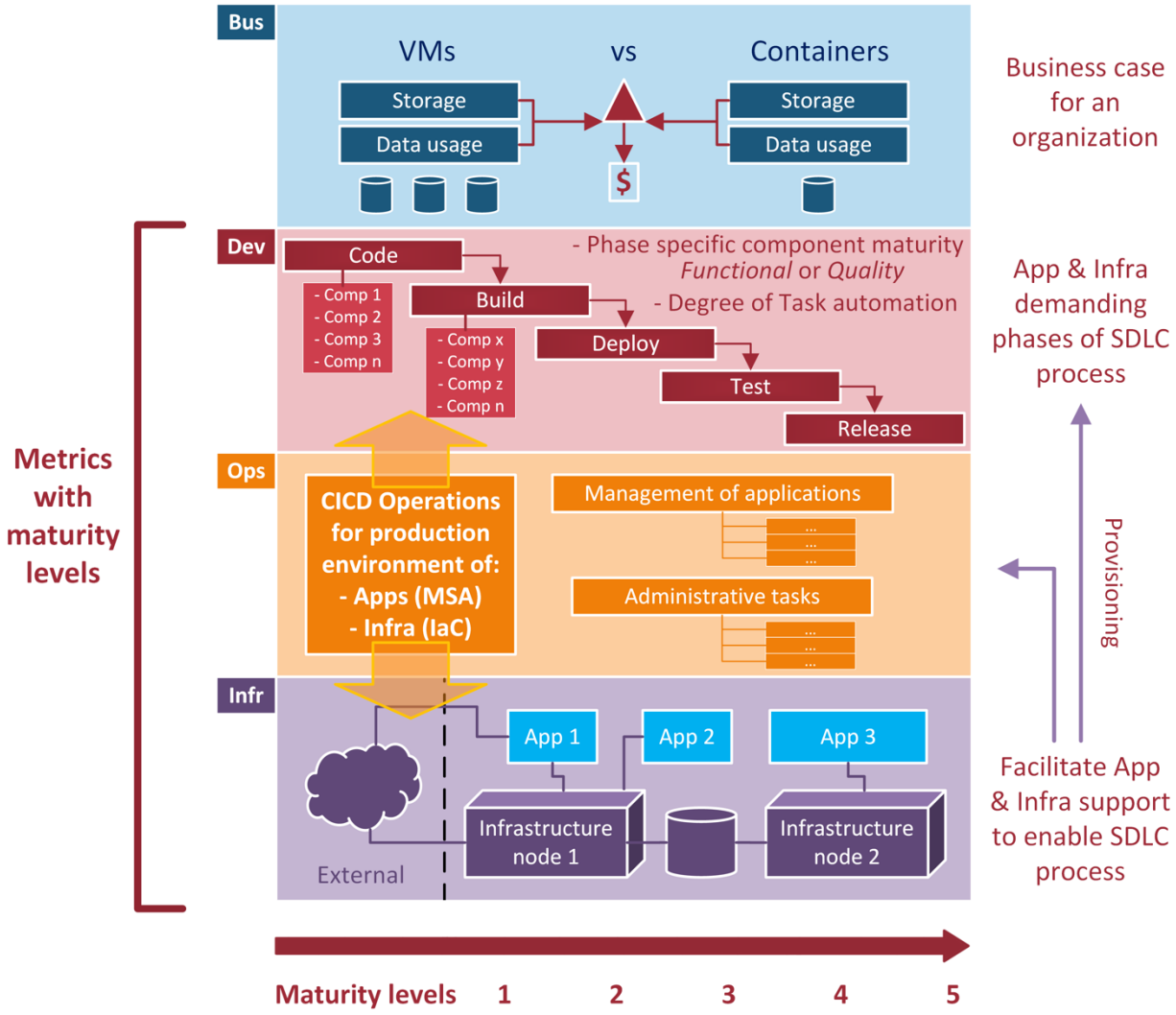


Figure 54: CMM MA area cohesion visualization

We chose to use the maturity level *names* of Dreyfus (1980), as we think these names are better suited to use in our CMM. Dreyfus denotes names for the learners (organizations in our CMMs case) of a skill and describes behavior per maturity level. Rather than the CMMI model, where the maturity level names lean more towards functional requisites – in that way restricting adopters of the model structure to a direction maturity – than describing behavior without limiting model adopters.

Whereas we describe in our CMM the functionality of containers with the focus on the SDLC process. We do not describe a process itself. Those level names can be used for process characteristics, and less for container functionality characteristics. Besides, the CMM does not focus on quantifying objectives.

However, as Figure 55 shows, we partially adopted the CMMI structure as the structure of our MA.

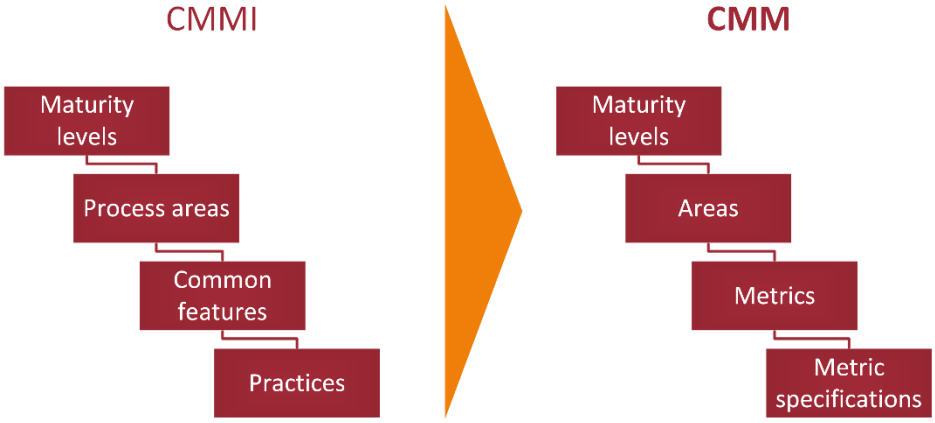


Figure 55: CMM adoption of CMMI structure

9.3.1. Business (output) area

The output of the model is intended to be generated automatically. It shows the potential savings that can be made by an organization if it changes from using VMs towards containers. The largest save would be to decrease the amount of VMs that are used to launch testing environments, and fastening the configure time of virtual environments. We found that savings can be made on several types of values (data, storage, IT budgets, build servers).

However, current calculations are too limited. The values are based on numbers we were able to derive from large cloud and infrastructure vendors. We understand that organizations can receive (a substantial amount of) discount when purchasing cloud or infrastructure resources (B2B). Therefore, we decided to exclude the content of this area. We did not want to include information of other quality than the remaining areas of the CMM. We propose that future research focuses on this aspect so that future developments can realize the business area.

9.3.2. Development area

Development (DEV) is the area that comprises of main SDLC phases, which are Code, Build, Deploy, Test, and Release. Throughout these phases, we arranged specific CICD components that deliver functionality. The type of functionality that such CICD components delivers defines maturity. For instance, a basic functionality is linked to a lower maturity level, and a more advanced functionality towards higher maturity levels. By doing this, organizations who have an advanced SDLC configuration achieve a higher maturity level than organizations who have a less advanced configuration.

Table 24: Pre assessment Areas and questions

SDLC	Metric	Description
Code	Code task automation	The extent of Code related tasks that are automated and integrated into a standardized pipeline. Common tasks are:
	Source code management system (code repository) – System specifications	Source code management system is a system that contains pieces of (source) code of an application or service. During the code phase, this system is used to transfer code to and derive code from. With 'System specifications' we mean the functional behavior or characteristics of this component inside the SDLC process.
	Source code management system (code repository) – Usage specifications	With 'Usage specifications' we mean how the component is used inside the pipeline.
	Base image standardization	A base image is a predefined piece of software containing executable code, network/API connection… We added this metric as containers are launched using base images. In more mature (formal) organizations, it is a best practice to use standardized, organization-wide obligated base images for different types of software.
Build	Build task automation	The extent of Build related tasks that are automated and integrated into a standardized pipeline. Common tasks are:
	Control	The metric that consists of controls that are performed before builds are executed. For instance, code-level integration controls or controls on all external dependencies of builds.
	Build server	A server that facilitates support to run builds that are being put together into a unit. The maturity levels include the way how builds are scripted, whereas scripts are the files that contain information to configure a build server.
	Artefact repository – System specifications	Artefact repository is a storage that contains artefacts. Artefacts are builds that have been put together from the Code phase. System specifications denote the functional behavior or characteristics of the component.
	Artefact repository – Usage specifications	How the Artefact repository is used within the pipeline.
Deploy	Deploy task automation	The extent of Deploy related tasks that are automated and integrated into a standardized pipeline. Common tasks are:
	Deploy container release	The manner how containers are released (i.e. launched) in the Deploy phase. This metric focuses on functional aspects, not on how container releases are automated.
	Deploy to state	A metric that denotes the different states an application can be deployed to.
Test	Test task automation (automated testing)	The extent of automation and integration into a standardized pipeline of different types of tests.
	Test environment managing	This metric is focused on how test environments are managed. This can be VM-based or container-based, and includes corresponding aspects like automation and decommissioning.
	Acceptation environment managing	This metric is focused on how acceptation environments are managed. This can be VM-based or container-based, and includes corresponding aspects like automation and decommissioning.
Release	Release task automation	The extent of Release related tasks that are automated and integrated into a standardized pipeline. Common tasks are:
	Release container release	The manner how containers are released (i.e. launched) in the Release phase. This metric focuses on functional aspects, not on how container releases are automated.
	Extent of controls	The extent of controls that are configured in in the SDLC process. These controls are performed before an application is released and brought live in production.

9.3.3. Operations area

The third area, Operations (OPS), contains all tasks that organizations perform after releasing their applications to production by using containers. These tasks are Monitoring, Version control, Lifecycle management, and Decommissioning. The manner of how these tasks are performed by an organization, indicates the maturity throughout this area. The main distinction between DEV and OPS is the fact that DEV refers to the non-

production environment where an application is in development (using containers), whereas OPS refers to the production environment where applications run (in containers).

Table 25: Pre assessment Areas and questions

Metric	Description
Automated rollback	Automated rollback is the functional ability in the operations process, that enables an organization to return to an older version of running applications.
Patching	This metric consists of the specifications on how patching of new software versions is configured in the operations process.
Application decommissioning	Denotes how live applications that are running in containers are taken down when not required by its users anymore. Maturity levels include the ability to (automatically) 'destroy' containers, and how information and resources are logged and transferred.
Version control managing	Comprises of functionalities that handle the management and maintenance of different versions of images and base images, and how they are connected to the SDLC process.
Monitoring	The level of detail of monitoring of applications. Monitoring can be focused on different levels, and exists in multiple states, which are processed into the maturity levels.
Logging	The Logging metric is focused on the extent of activities that are logged throughout both the SDLC and operations process.

9.3.4. Application and infrastructure area

The application landscape and infrastructure provision an organization's business operations, hence the SDLC and operations process. The hierarchy of the CMM we structured is comparable with widely-known layers of the EA discipline: (bottom-up) infrastructure (Infr), application (App), and business (DEV and OPS).

Table 26: Pre assessment Areas and questions

Metric	Description
Business logic container usage	This metric is a higher-level metric than other metrics. It describes how business logic is positioned in the SDLC and operations process. This includes aspects around MSA.
Load balancing	Load balancing denotes how demanded applications and services are balanced between application landscape elements and infrastructure elements.
Scaling of applications & infrastructure	Contains maturity levels that show different ways of scaling applications and services by the support of infrastructure. It describes the relationship between the application landscape and infrastructure, and the extent of automation with management tools.
Resilience testing	Different types of tests around the resilience of the application landscape and infrastructure. Through the maturity levels, we described what kind of monitoring is performed, and subsequently what activities with different level of complexity are executed to make the landscapes resilient.
Container failure handling	Focuses specifically on the containers in the infrastructure. The metric indicates how container failures are managed and to what extent recovery of containers is automated.
Connection to persistent storage	As the main application of containers are providing environments suitable for stateless applications, there is on persistency. Therefore, rapid established connections to persistent storages are important for container utilization. This metric incorporates the varying levels of multiple ways to connect to persistent storage.
Service orchestration	Requested services have to be launched in containers. However, before such a service is assigned to one or more containers (infrastructure elements) in order to run on it, orchestration tools determine the configuration.
SDLC pipeline container usage	Another higher-level metric that is focused on overall container usage throughout the SDLC pipeline. It is possible that organizations want to migrate to containers in small steps, instead of a big bang migration. Therefore, we included this metric to show organization what SDLC phases should be containerized first.

9.4. Results matrix

The result matrix shows the results of the maturity of an organization. Per area, topic, and aspect, the maturity score is given from which one can derive and select its topic(s) of excellence and topic(s) of improvement. With this information, organizations can orientate themselves on improving their overall SDLC maturity in regards to containers. Moreover, based on the Results matrix, a customized advice can be provided in order to start with containers, or to further utilize containers in the SDLC process.

The CMM assesses an organization's container-readiness, and the maturity of their SDLC process, whereas the latter also focuses on the application landscape and infrastructure, and corresponding business value. Different areas consist in the CMM, whereas each area includes several metrics. These metrics are components that are used in the SDLC process, and are defined throughout five different levels of maturity. Indicating an organization's maturity for each metric provides the organization with an overview on their SDLC configuration performance. With this knowledge, an organization notices how it currently performs, and can set goals towards higher maturity levels. These higher maturity levels are defined into metric specifications, and are noticeable by the CMM. Therefore, with the current 1.0 version of the CMM, an organization can inform itself by looking into the model and decide what its new aimed for maturity levels are.

9.5. Container Maturity Model – Disclaimers

Throughout the CMM, we apply the statement that containers are the preferred choice besides VMs. We based this statement on our research throughout the document, where we proved that SDLC is a use case that benefits from using containers.

This model is also applicable to the execution of VMs, besides the focus on containers. It states the same kind of usage requirements, agile operations handling, and output format. The only difference is the used technology and its corresponding behavior, which to different performance and other way of operating the environment.

When an organization is already advanced with the use of VMs, it is likely that the transition to containers is rather simple and results in immediate high maturity level results. As aforementioned, it is only the technology that is different, whereas the configuration of operations can be identical. In theory, the transition does not have to be that large.

10. Results: Process Deliverable Diagram of CMM

The CMM contains multiple phases where different parts are in performed. These parts are dependent on each other's input, in order to initially start the follow-up phase. Hence, the phases must be executed in a specific order and it should be guaranteed that the required deliverables are realized. At first, the PA should be executed, followed by the MA, and finally evaluating the Results matrix. The complete sequence results in different sub-products that form the main *business case* together.

The Process Deliverable Diagram (PDD) meta-modeling technique of Van de Weerd and Brinkkemper (2009) is used to visualize the CMM's sequence and deliverables. A PDD combines two UML diagrams: 1) UML activity diagram (left side), and 2) UML class diagram (right side). Both sides are connected to each other by denoting what (set of) activity(s) realizes a specific deliverable. The PDD elaboration on the CMM is given in Figure 56.

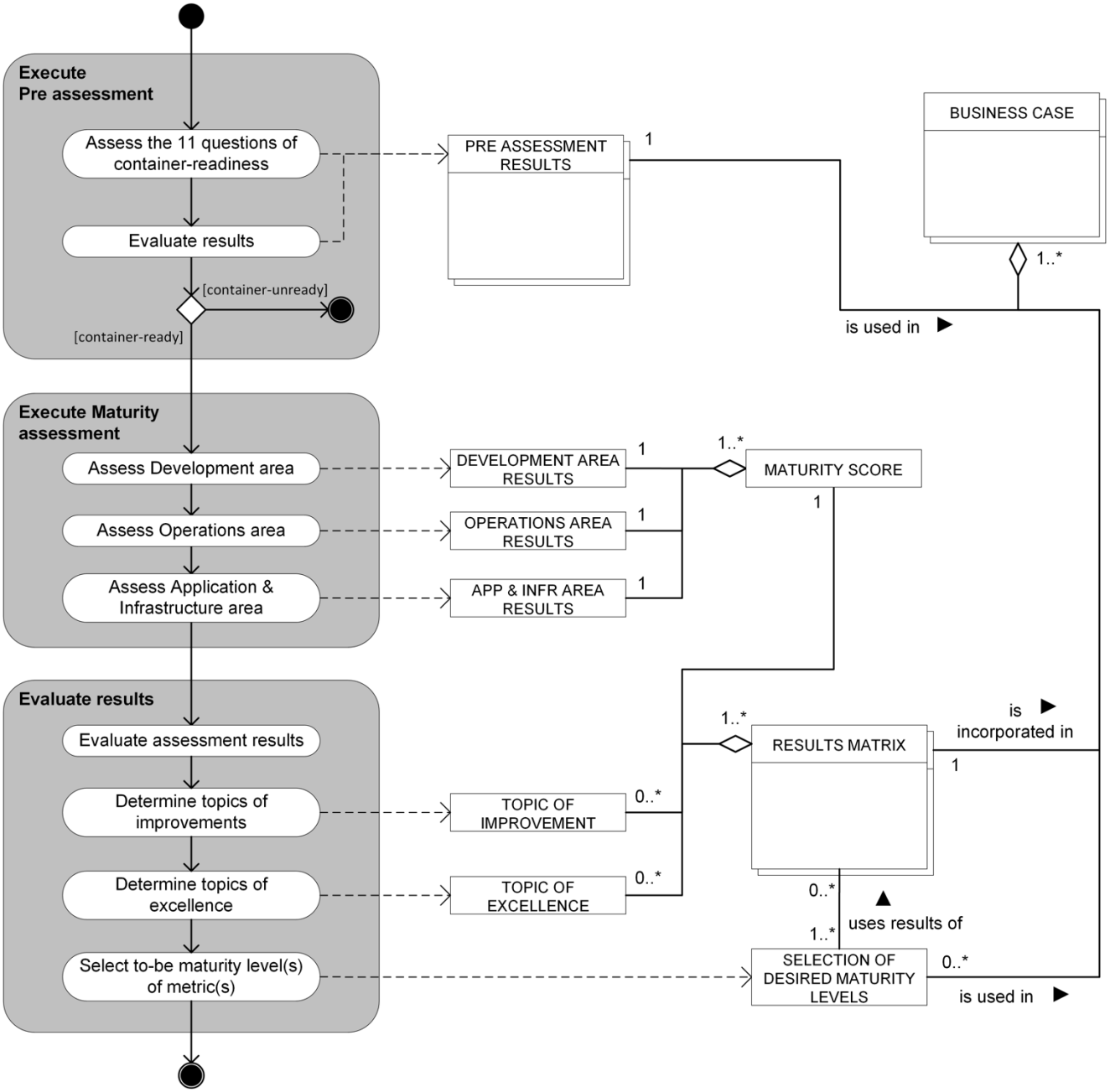


Figure 56: CMM processed into a PDD

By following the PDD guidelines of Van de Weerd and Brinkkemper (2009), we created two tables describing the PDDs activities and concepts. These tables are given in Appendix X.

10.1. PDD Phases

10.1.1. Phase 1: Execute Pre assessment

As aforementioned, the PA determines whether an organization is container-fit, thus can continue executing the MA. Determining the container-readiness is done by assessing the organization with 11 questions. Subsequently, the results of the questions are evaluated, deciding about the container-fitness. The outcome of the 11 questions, and decision on an organization's container-fitness form the Pre assessment results. The concept is used in the Business case to facilitate a brief overview about an organization's overall technology state. This state considers the three areas of the PA, which are General, Software development, Applications and infrastructure.

10.1.2. Phase 2: Execute Maturity assessment

The MA is takes more time to complete than the PE. Main activities are completing the assessment regarding the areas of Development, Operations, and Infrastructure. The Business area is generated afterwards. The output of all areas combined, form the Maturity assessment results, and are shown in the Results matrix.

10.1.3. Phase 3: Evaluate Result matrix

The final phase comprises evaluation aspects. Here, the Results matrix is analysed, wherefrom topics of excellence and –improvements are distilled. These topics together form the Maturity score. Based on these results, an organization can determine what topics or metrics it will focus to improve their SDLC process maturity in regards to containers.

10.2. PDD contribution

Current PDD elaboration regarding the CMM is a first concept of how the CMM process should executed. We provided a base of the sequence and corresponding deliverables of the CMM.

Further on in this document (section 15 Discussion and future work), we propose several improvements for all three parts of the CMM that should be further investigated as future research. When these improvements are implemented into the CMM, we believe that these improvements bring impactful changes to the CMM process- and concept side. For instance, if the proposed interrelationships between metrics are found, the third phase should be modified in order to incorporate the specifications of the improvement.

Besides that, we also believe that other improvement will emerge when more research is conducted towards the CMM. It is possible that these contributions will lead to modifications in the MA, PE, or Results matrix, and implement new additions to metrics and topics. Moreover, this can additionally influence the overall structure or sequence. Either way, activities and concepts are likely to be changed over time, or further developed into (more) mature variants.

Considering these possibilities and the likelihood of all changes and additions, we believe that providing a PDD of the base process of the designed CMM realizes two aspects. First, the PDD supports users in using the CMM in practice, as it clarifies the CMM's sequence. Secondly, future CMM developments can be processed faster into the sequence, as the PDD provides an overview of the structural phases, activities and deliverables.

11. Results: The CMM linked to EAG

As aforementioned in section 1.1 Introduction and 3.4 Enterprise agility, the core agility attributes that are applicable to OPT are 1) Flexibility, 2) Speed, 3) Responsiveness, 4) Low complexity and Integration, and 5) Culture of change according to Sherehiy et al. (2007). We linked VMs and containers to these core attributes (culture of change excluded) to denote what attributes are in our opinion enhanced by functionality provided by VM technology and container technology. By doing this, we linked containers to EAG on conceptual level, therefore showing that containers are an agile enabling technology.

In order to show how our designed CMM is related to EAG, we linked the CMM content towards the same core agility attributes. By grouping the metrics into several topics, we linked these topics with the core agility attributes in order to show how each topic fosters EAG on detailed level. Table 27, 28 and 29 denote the established links between topics of the CMM with agility attributes. Furthermore, we provided in Appendix XII additional explanations on how we consider the CMM is connected to the attributes of EAG.

Table 27: Development topics

Topic	Metric	EAG attributes
Task automation	Code task automation	Speed Responsiveness
	Build task automation	
	Deploy task automation	
	Test task automation (automated testing)	
	Release task automation	
Component maturity		
- Functional	Source code management system – System specifications	Flexibility
	Base image standardization	
	Control	
	Build server	
	Artefact repository – System specifications	
- Quality	Source code management system – Usage specifications	Speed
	Artefact repository – Usage specifications	
	Deploy container release	
	Deploy to state	
	Test environment managing	
	Acceptance environment managing	
	Release container release	
	Extent of controls	

Table 28: Operations topics

Topic	Metric	EAG attributes
Lifecycle management	Automated rollback	Speed
	Patching	
	Application decommissioning	
	Version control managing	
	Monitoring	
Administrative tasks	Logging	Integration and low complexity

Table 29: Application landscape & Infrastructure topics

Topic	Metric	EAG attributes
Landscape resilience	Resilience testing	Flexibility
	Container failure handling	
Service management	Load balancing	Integration and low complexity
	Service orchestration	
Network maturity	Connection to persistent storage	Responsiveness
Scaling potential	Scaling of applications & infrastructure	Responsiveness

Treatment Validation

12. Validation of the CMM

In order to increase the validity of the designed artefact, we validated the CMM's content. We realized this through the expert interviews from IT2, IT3, and IT4. This means that we combined the TD phase with the Treatment Validation (TV) phase. Next sub-section explains why we made this decision.

12.1. Reporting on validation

In the design plan, we defined four different iterations to execute. Each iteration was focused on gathering knowledge from a different type of source (literature or experts), to subsequently process the rationalized findings into the CMM.

Since the domain of SDLC was relatively a new topic for us, we created a design plan where we collaborate with experts to design the model. In IT1, we aimed to design the first concept version of the CMM. Whereas in the following iterations, we discussed the content of the CMM with experts (validation) and analyze their feedback and perspective. Subsequently, we rationalized the feedback and processed it into the CMM (designing). Due to this collaboration, the experts had an important role in our design plan. Consequently, we considered the opinion of experts as a high value for the model. Therefore, the interviews had two main objectives: 1) gathering knowledge to further design the CMM, and 2) finding out the expert's opinion on the presented content of the CMM, in order to validate the concerning concept version.

Finally, to measure the effect of the CMM in the problem context, we would use the resulting CMM concept version of IT3, in the fourth iteration to conduct case studies at external organizations. However, we noticed that the CMM's current state was not fully sufficient to use in case studies. This had the following reasons:

- The CMM was continuously under change (i.e. still in development), based on the differentiating feedback of all expert interviews;
- Several experts required additional explanations around metrics, before they understood what was meant. These request varied, which was depending on the specialization of the experts.

Considering the observations of the CMM's continuous change and the requests for metric explanations, we concluded that the CMM was not ready to use in case studies. Consequently, and based on these observations, we determined that we needed further designing and validating the CMM, in order to improve the content and increase its validity.

Therefore, we changed IT4 by conducting additional expert interviews to design and validate, instead of case studies. We maintained the same scheduled interviews, managed to schedule extra expert interviews, while using the interview protocol of IT3 with slight adjustments. Figure 57 depicts the changed situation.

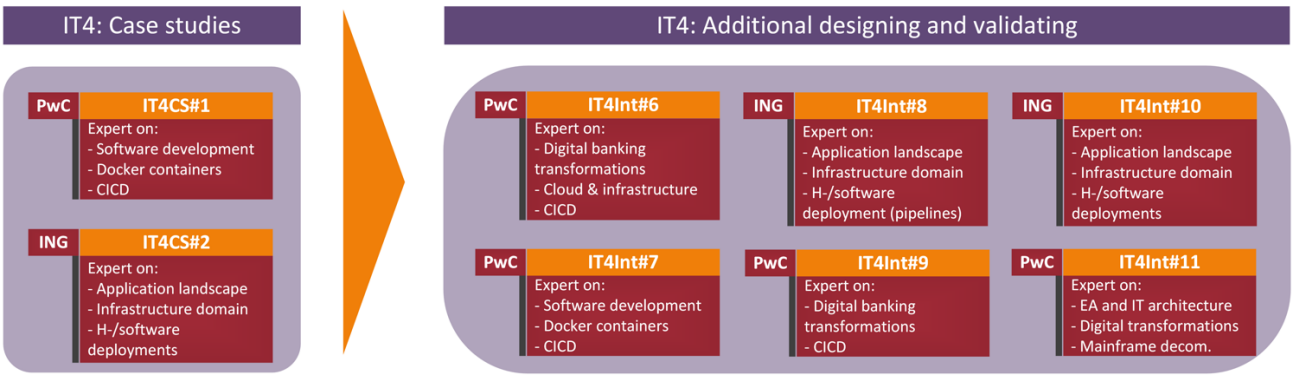


Figure 57: Treatment validation iteration change visualized

Hence, by evaluating the situation and changing the iterations, we provided a base version of the CMM that can be used in case studies as future research. However, we were not able to perform a case study for the model ourselves, as we had a limited period of time available and all the continuous changes required this time to process. We believe that this step increased the CMM's overall validity in reference towards the real world.

13. Reflection on Research Process

This section provides a reflection on the research process. We focus on the challenges we experienced during the research process, and on quality of research.

13.1. General reflection on process and results

13.1.1. The beginning

At the moment the short proposal was delivered, we were not an expert on any of the involved concepts. In addition, none of the involved parties had an idea what form the resulting artefact would be. We used the first months of the PI phase to study and understand the main concepts. We believed that, besides defining the theoretical foundation, this also supported the process of discovering what kind of artefact we could design to interact with a problem in this context.

13.1.2. Our challenge around containerization

During these first months, as we did not find any literature that linked EAG to containers, we also were not able to discover any challenge regarding the usage of containers. The only denoted issues we found in the literature and by desk research, were the weaknesses of containers regarding their security. However, designing a solution for this problem was not feasible for us. This security issue is located in the kernel of an OS where containers are connected to, which requires us to explore containers on code level and write software. That goal did not align with our background and possessed skills, and was therefore out of scope for this project.

Another option was to conduct research on the traceability between the transition from VMs towards containers. We hypothesized that VMs possess (functional) characteristics that have to be taken over by equivalent container (functional) characteristics. In this way, we could explore how both VMs and containers differ from, and share resemblance with each other on functional base. We investigated the literature and did not find existence of such knowledge about the relation between both concepts, meaning that this option had relevance to further investigate. However, when we proposed this to experts from the field of practice, they argued that other aspects and cases would be more relevant to develop.

We found through expert interviews that organizations have a need for knowledge on 1) starting to apply containers or 2) further improving the usage. Some organizations may know the advantages of containers, but do not have in-house knowledge on containerization to utilize it, or do not know how to improve or mature its utilization. This was the first indication of a suitable challenge around containerization.
In addition, the experts stated that for organizations, SDLC is one of the most important use cases for applying containers. Furthermore, they argued that organizations would benefit if they had access to knowledge on applying containers with different levels of skill.

We concluded that this challenge was more of conceptual nature, rather than the isolation issue. This challenge could be treated by designing a maturity model that provides knowledge on such differentiating levels of skill in respect to containers. Moreover, these levels provide knowledge to both calls from industry, as mentioned by the experts. By combining the denoted call from industry for differentiating knowledge on container utilization, and the important use case of SDLC, we distinguished a relevant and suitable challenge to explore and conduct research on. Therefore, we decided to design the CMM with the focus on SDLC.

13.1.3. The design phase

We had to extend our literature review by investigating SDLC. Conducting additional research required extra time as SDLC relates to other new topics, which we included in the literature review. Conducting research on SDLC resulted in many articles about SD and SD-methods. This applied for both literature as desk research sources. Therefore, we followed the literature and kept focusing on the development process of SDLC. We also had to consider that the addition of SDLC might lead to the research becoming more of a technical essence. However, we evaluated this risk with all involved parties, and considered conceptual knowledge on the concepts as sufficient to design the CMM.

We wanted to collaborate closely with experts to design the CMM, as we needed more knowledge on SDLC. Once we started the expert interviews in the TD phase, we expected a large amount of feedback with a high information density, coming from different perspectives that together ultimately enhanced the validity. Especially since the experts' specializations varied from developers, architects, and implementers with seven years of experience in IT on average. However, the provided feedback did not contain large comments or major suggested changes. We did receive many suggested additions to the model on metric specific level, but we had to ask specifically to each metric.
Moreover, we thought that interviewing experts with varying specializations would contribute to the knowledge intensity we gathered. However, we found that this variation realized in the CMM being constantly subject to change.

We considered both as a threat to validity, since it was unlikely that our designed model would be close to reality through the first iteration. Next sub-section elaborates on this threat. Nevertheless, from out this observation we concluded that the CMM was in need for more development, which could lead to shortage of time or a not (fully) tested model.

13.2. Research quality

13.2.1. Threats to validity

During the expert interviews of the TD phase, most experts were initially not as critical as we expected. When we continued to ask specifically for their opinion regarding each metric, we received more feedback around the topics. We believe that this is because of the short time experts were available. Considering this time challenge, and to increase the chances of being able to interview them, we restricted the interviews to one hour (except for IT3Int#5).

Besides that, we experienced that one hour was insufficient to discuss the complete model with experts. Hence, we were forced to take action by interrupting experts' explanations about topics, or skip parts of the model. We made sure that we would discuss skipped parts into subsequent interviews. This was, however, not an ideal situation. During the later interviews (IT4), we separated them into multiple sessions to counter this situation.

Another related challenge was the continuously changing CMM. This challenge, however, was both negative as positive. On the negative side, we were not able to start with the case studies of the model within the time of our project. However, at the other hand, due to all the changes from the varying feedback on both CMM's content and structure, we believe the model has become more valid towards the real-world context. Therefore, improving the results of this research.

We planned to test our CMM through case studies at two large organizations. Since we noticed that the CMM was not ready for validation, we decided to change the activities of IT4. We believe that because of this decision, current version of the CMM is more complete than it would be by keeping the original plan. Therefore, we believe that current CMM version can be used in case studies as future work.

14. Conclusion

In this section, we conclude our research in this master thesis project, where we aim to link and explain our answers to the defined RQs of section 1.2.2. After each summarized explanation about a topic, we denote which RQ we answered.

14.1.1. The all-encompassing domain of enterprise agility

Nowadays, technology is involved with almost any change in an organization. Especially in business transformation and -digitalization, new technologies are introduced. These new technologies should be compatible with all current systems and applications of an organization to be able to implement successfully. Depending on its magnitude, the new technology affects both business, application and infrastructure domains. This means that such changes start with investigating the enterprise and IT architecture(s) of an organization.

We aimed to use containers – a specific technology inside the virtualization technologies domain – to improve an organization's EAG. Containers provide organizations with the ability to change their current way of supporting, deploying, and maintaining both hardware and software. Besides that, containers have characteristics that align with the characteristics of EAG, baptizing containers as an 'agile-enabling technology' in our research. In most cases, an organization can apply containers on positions where VMs are currently used. However, VMs align less with EAG in equivalent contexts. To depict this difference in regards to EAG, we linked containers and VMs to EAG in the introduction of this thesis.

Subsequently, the research continued in discovering what the literature and experts from the field of practice write, think, and experience in reference to agility, EAG, and their link to (agile enabling) technologies, including the virtualization technologies. From the literature, we found that multiple agility types exist, which all focus on either Organization or People of OPT. However, we found none that referred specifically towards Technology (section 3.4). Consequently, from the literature we were not able to find a specific link between both EAG and technology, nor EAG and containers. Secondly, all experts we interviewed during the PI phase agreed that a container is an agile enabling technology, acknowledging its potential to improve EAG regarding technology. Furthermore, these experts denoted that the biggest challenge around containers for organizations is the lack of knowledge on how to start applying containers or further utilize them. Subsequently, we discussed and concluded that the field of practice would benefit with a model that provides organization specific 'hands-on' knowledge to start applying containers in their SDLC process. Hence, based on the found *gap of knowledge* from the literature and *need for knowledge* originating from the field of practice, we decided to design a container maturity model that would provide an organization with insights on utilizing containers. Ultimately, in order to support their enterprise agility from a technology perspective.

*As we reported about the SOTA of agility, EAG, virtualization and containerization, included the opinion of experts from the field of practice, and linked both VMs and containers to EAG, we answered **RQ**1.1, 1.2, and 1.3.*

14.1.2. The extension of literature reviewing

We determined that additional literature reviewing to SDLC and maturity models was required, after the decision to design a container maturity model. In addition, we also had to find and define how containers can be integrated into an organization's EA. As a result, we studied different maturity models and selected a structure and specific content to adopt in our CMM. Furthermore, we investigated the SDLC domain to find out what it comprises. With this additional reviewing, we aimed to become known with the new topics and extend our theoretical foundation. Consequently, we were able to design the first concept version of the CMM (v0.1), which we used to assess through expert interviews of IT2.

*As we further investigated SDLC and maturity models, and subsequently stated our results, we answered **RQ**2.1.*

14.1.3. Containerization in enterprise architecture

Exploiting the advantages of containers is use case specific. Especially short-lived, stateless instances of applications or services benefit from containers. MSA is a pattern that decouples applications into decoupled services (i.e. migrate from monolith towards independent services), realizing services to be individually

modified, executed, and maintained. An organization that uses the MSA pattern in their EA has the architectural requirements to benefit from using containers. Moreover, the IaC principle is also enabled by this pattern, as containers are infra-agnostic, and can be easily configured by using configuration scripts and infrastructure files. The combination of these four concepts together enable an organization to become more agile throughout the whole enterprise, specifically by the means of technology.

Hence, we found the following synergy between conceps and technologies: the MSA pattern realizes the use case-specific demands that containers require to exploit their benefits (portability, stateless, resource efficient) in an EA. Therefore, integrating containers in an organization's EA while applying the MSA pattern increases EAG in regards to technology, and enables IaC, which further enhances EAG.

*We studied the functionality of containers, and subsequently used the results to find a suitable manner to implement containers in an EA. By the combination of the four technologies (MSA, IaC, EA, and containers), the benefits of containers are enabled in an organization's EA. Hence, we answered **RQ**2.2.*

14.1.4. Designing an artefact for the problem context

We chose to use multiple iterations to design the proposed CMM. Each iteration had a different setting and goal, where we aimed to collect varying types of information. IT1 activities consisted of brainstorm sessions and prior PI literature results, whereas the remaining iterations were focused on semi-structured expert interviews which also included validation. This collection of varying information acted as our main source to design the CMM from. Hence, all iterations combined depict a design process of incremental nature. We believe that the different information perspectives (as described in our Design plan in section 7) of this design process, enabled us to deliver a sufficient first version of the CMM that can be used to encounter the problem context. Moreover, this CMM version is ready to be further investigated through future research to strengthen its validity and effectivity.

*Our constructed design plan shows an incremental design process which incorporates varying types of information originating from multiple perspectives. Hence, we partially answered **RQ**2.3. This RQ is completely answered with the final result of the CMM, as described in the next sub-section.*

14.1.5. The designed artefact and main RQ

The CMM assesses an organization's container-readiness, and the maturity of their SDLC process, whereas the latter also focuses on the application landscape and infrastructure. Different areas consist in the CMM, whereas each area includes several metrics. These metrics are components that are used in the SDLC process with respect to containers, and are defined throughout five different levels of maturity. Indicating an organization's maturity for each metric provides the organization with an overview on their SDLC configuration and performance. With this knowledge, an organization has an indication how it currently performs, and can set goals towards higher maturity levels related to containers. Therefore, an organization can inform itself by using the model to determine what its improvement strategy for achieving higher maturity is going to be by following our metric specifications of each metric. Returning to our main research question:

How can organizations support their enterprise agility regarding business and IT by integrating virtualization technologies?

We answered the main research question, as organizations can use the CMM to derive the current state of their SDLC maturity regarding containers, and can set a strategy for new targets based on our defined maturity levels. Besides that, the container technology is the mean in this case. However, before we can apply this mean into an organization, its architecture and corresponding applications have to be compatible. Without that, containers will not bring the promised agility, despite their characteristics. A MSA pattern helps in such case, as this decouples applications and system into small, independent services or individual pieces of software. Moreover, MSA also enables the IaC principle, further enhancing enterprise agility in an organization. Hence, the combination of the mean of containers, the fundament of the MSA style adopted in an EA, enables IaC and lets the organization use the CMM to further increase the maturity of their SDLC process. The higher the maturity, the easier it gets to advance in enterprise agility.

14.1.6. Validation of the CMM

We decided to change the case studies of IT4 to extra expert interviews in order to gather additional knowledge for designing the CMM and to validate the presented version. Based on the response of the different experts from the field of practice, gathered in IT2, IT3, we concluded that the CMM was still subject to change during the final iteration. Therefore, we did not start using the CMM in case studies, but gathered additional knowledge instead to further design, improve, and validate the CMM.

*Despite we changed the activities of IT4, we were able to provide an answer to **RQ**3.1. RQ3.1 is focused on how experts from the field of practice perceive the CMM in its assessed state.*

14.1.7. Our contribution to the scientific domain and field of practice

As aforementioned in section 3.5, we were not able to find explicit links between EAG and a functional technology. Nevertheless, the four characteristics of agile IT we did find are relatable to our answer on the main research question. By using MSA with containers, we realize 1) *Technical integration* of the container technology within an organization's EA. Subsequently, by increasing the maturity of the SDLC process, i.e. increasing the formalism of the process (as found in section 3.10.4), we reach a higher 2) *Standardization*, as more components are standardized and defined. High standardization leads to increased chances of 3) *Reusable* components of any other aspect from the SDLC process or delivery. As the final one, by using containers and independent, decoupled services, a high 4) *Scalability* potential is enabled in the infrastructure and application landscape of an organization.

Furthermore, we believe that most organizations that develop software are familiar with the CMM metrics we have defined. The metrics originate from generally accepted CICD sources, SD and infrastructure experts, and SD best practices. Due to organizations' familiarity with CMM content, we argue that users are able to align the metrics with their organization-specific context, and therefore (partially) understand what the next maturity level for a metric comprises. Besides that, most of the larger organizations already apply virtualization by using VMs and hypervisors, and are therefore familiar with virtualization technology related components.

Achievements towards higher maturity levels improve small parts of the organization's SDLC process. For each topic, we made a link towards the EAG characteristics (as we also did for virtualization technologies in section 1.1), as a metric or metric specifications provides either new functionality or quality improvements. This means that advancing a maturity level onto one or more metrics, incrementally contributes to an organization's overall EAG regarding Business and IT – the T (technology) of OPT. Consequently, we can reject the null hypothesis (H_0), and partially conclude from mainly a theoretical perspective that the alternative hypothesis (H_1) is true.

In addition, we provided a PDD of the CMM to facilitate an overview of the CMM's sequence and corresponding deliverables. With the PDD, users or researchers are given an overview of the CMM's structural phases, the sequence, and deliverables. By doing so, the PDD supports usage of the CMM, and also future developments by making it easier to implement.

Concluding, organizations using the CMM v1.0 are provided with insights that is usable for the improvement strategy of their software process maturity. This supports their EAG regarding Business and IT by integrating virtualization technologies – more specifically: containerization or containers.

15. Discussion and future work

15.1. Virtualization technologies and related technologies

15.1.1. Security of containers through architecture

Strengths and weakness of containers have been explained in this MSc thesis. Containers' major weakness is the lack of isolation – due to the shared kernel (as visualized by black arrows in Figure 58) – restricting containers from being used in use cases where security is essential. To fix this issue, two main options are available:

- Further research on containers to improve isolation;
- Hybrid architecture solution of putting one or more containers inside a VM.

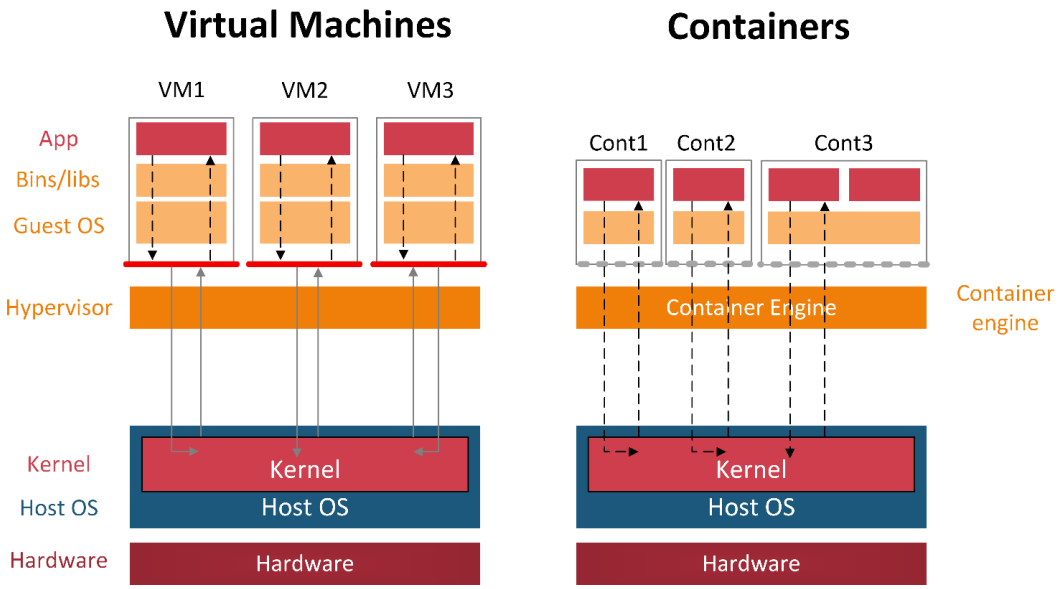


Figure 58: VM and container kernel isolation

Currently, both options are investigated by scientists (Arnautov et al., 2016; Tesfatsion et al., 2018; Bui, 2014). As our research shows, VMs have increased progression in development and can additionally be complemented with new concepts (examples given in next sub-section). We think it is wise to include VMs in further research as well, as the architectural combination of VMs and containers is currently popular in the business domain. Moreover, it also shows further potential due to new concepts that can be added to VMs and containers, creating a new synergy between both technologies (last two architectures in Figure 62, section 15.1.3).

15.1.2. Dynamic OS images

We have designed a maturity model specifically for containers. However, VMs have also progressed in their development. New OS images have been developed to improve performance of VMs. As an example, there is the term JeOS: 'Just enough OS'. This resulted in the concept of *unikernels*. A unikernel can be seen as a 'lean' OS image where only the essential parts that are required by an application that is aimed for are implemented. Another concept also emerged besides the unikernels, which is a concept of m*odular and stateless* OS images (example of provider is Clear Linux). Instead of making an OS more lean, this concept complements OSs with functionalities in order to prevent the need of having to launch another VM. See Figure 59 for a visualization of both concepts.

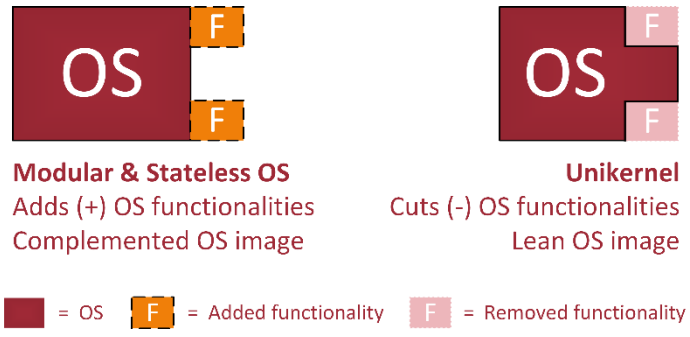


Figure 59: Unikernel and Modular OS concepts visualized

In Figure 61, a use case from practice is depicted. The unikernel's architecture needs two VMs,

whereas the modular image adds a functionality required by the second application to run. This addition removes the need for another OS, and is therefore able to fork both containers in the same VM.

Using such a unikernel or modular OS image for a VM instead of a full OS image is likely to save computing resources and increase speed. Combined with a container, these new architectural concepts can make use of both the benefits of containers, as the isolation functionality of VMs, but now at a 'fraction of/reduced amount of' performance penalty/overhead that accompany VMs. We think it is interesting to conduct further research on how the popular hybrid solution can be further advanced using different technologies.

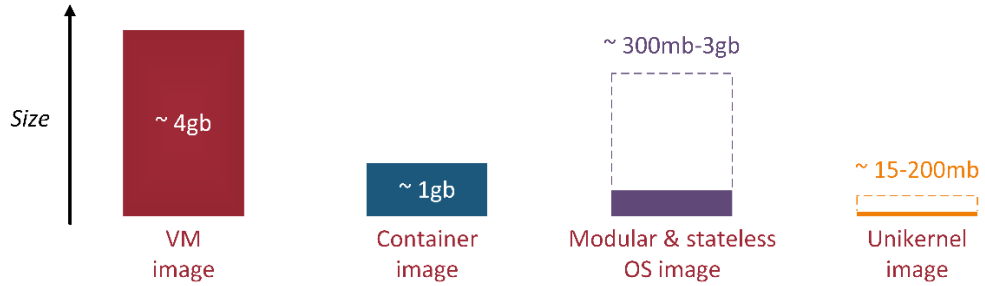


Figure 60: Image hierarchy (rough estimates of averages)

15.1.3. Hybrid solutions

As aforementioned, many organizations apply hybrid solutions (containers nested in VMs) in practice nowadays. However, when the hybrid solution is complemented with, for instance, the unikernel concept, the overall architecture becomes leaner than before. Theoretically, by doing this, it reduces the authentic VM performance penalty, maintains VMs' isolation benefit, and exploits the functional advantages of containers. We believe that further investigating and maturing the hybrid concept, combined with new technologies such as unikernels, will be the future for flexible and secure infrastructure. Consequently, it enhances container usage in new use cases, besides the cases containers already have proven their advantages.

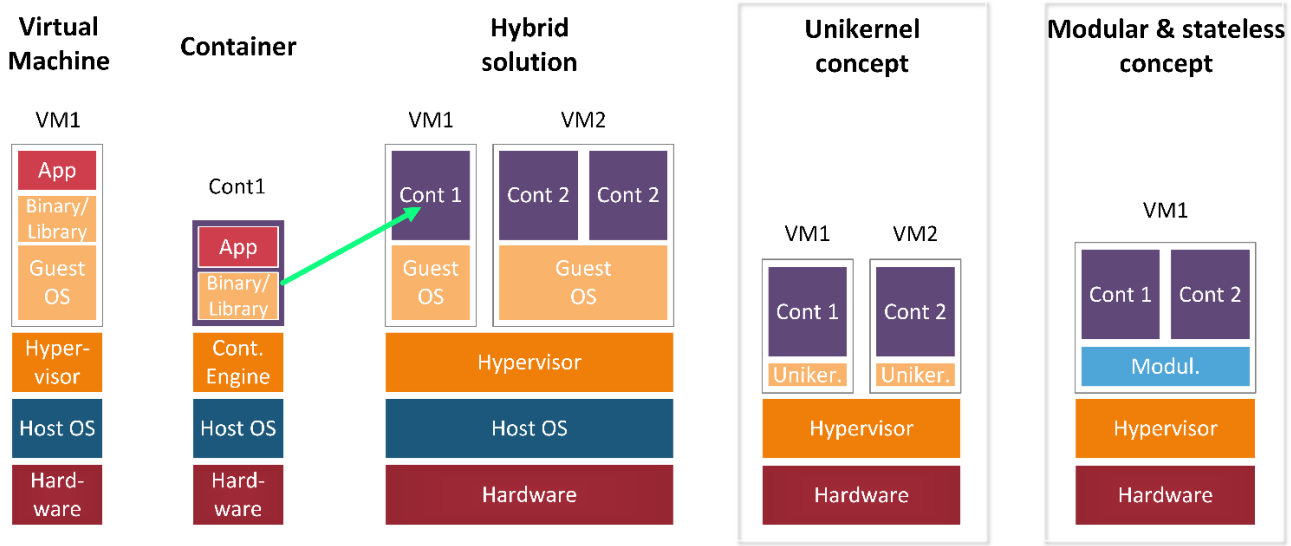


Figure 61: Virtualization architectural differences

15.2. Next developments on CMM

15.2.1. Cross-links of implemented CICD metrics

The current state of the CMM indicates the maturity of an organization regarding their SDLC configuration in relation to containers. If an organization has the need to switch from VMs to containers, it can use the CMM to test whether they are fit for containers, and to find out on what topics they should start focusing. Considering this information, the CMM fulfils its task as was defined by the need of organizations found during the Long Proposal. However, the CMM in its current state does contain untapped potential that requires additional

research.
CICD components of each SDLC phase have different interrelationships with each other. We think that having certain CICD components enabled in a SDLC configuration, it is likely that this also enables other components to exist more easily or improve their functionality. For instance, having a source code management system that is integrated in the journey fosters the ability to have automated builds. Hence, realizing untapped potential ('quick wins') organizations may possess (Figure 62).

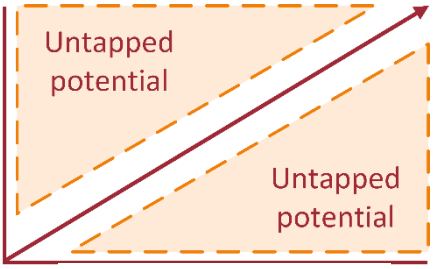


Figure 62: Untapped potential

Above example of such *cross-link* implies there are more cross-links hidden between multiple CMM metrics. Knowing those underlying cross-links enables researchers to further develop the CMM, by using the interrelations to provide a more accurate and realistic advice. Ultimately, this would improve the third part of the CMM by complementing Results matrix with a valid 'Roadmap' that contains organization-specific information to improve about that organization's SDLC configuration. The information is shaped organization-specific, by using the results of the PA, MA and underlying cross-links and interrelationships. Such a roadmap includes information that enables both indicating and achieving quick wins, as avoiding irrelevant (quick) wins, or ignoring slow incremental improvements, and unnecessary wins against high expenses (as shown in Figure 63).

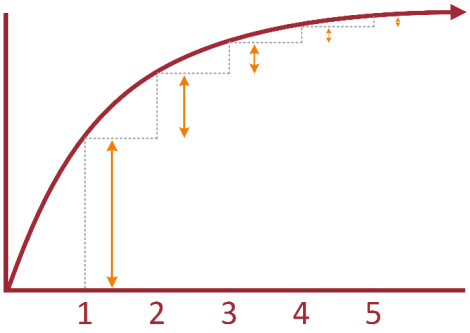


Figure 63: scenario of realized improvements for advancing maturity levels

Therefore, we state that further research should be focused on finding the interrelationship between the CICD components inside the SDLC process in regards to containers, which are processed into the CMM as metrics and as specified maturity levels.

15.2.2. CMM – Business area: output

For defining the numbers of all values used in the Business Area, we had to base our model on the given prices by vendors of such services. However, as our current calculations are limited, these values are not realistic for businesses, and other work prevailed before this area, we decided to exclude the content of the Business area. Moreover, it is likely that organizations (B2B) can receive discount or other payment constructions that were not visible during our research period. We propose that future research should focus on deriving more realistic values from relevant organizations, in order to improve the validity of the Business area and bring the results closer to the real-world context. Rather than including information of other quality than the overall CMM.

15.2.3. CMM – Roadmap

In current CMM version, the third part consist of the results matrix that depicts the maturity score of an organization. For future research, in collaboration with the investigation to the cross-links of CICD, we believe it is interesting to conduct research on creating a Roadmap related to the Results matrix. Such a roadmap shows the optimal for an organization to perform a specific (chain of) metric improvements based on the individual context of that organization. Therefore, the advice to improve is not distilled from the topics that can be improved the most, but rather the topics where that specific organization benefits the most from, all in relation to the organization's current SDLC configuration.

15.2.4. Validation

As aforementioned, the original research plan was to validate the model in the final iteration (IT4). Due to model's continuous changes based on expert interviews, we did not perceive the model as sufficiently advanced to start validation sessions (as explained in section 12). By changing the validation sessions to additional knowledge gathering expert interviews, we were able to design a further advanced model. We believe that this model ready to start with validation sessions, as the model would be without changing the research plan schedule. Therefore, we state that for future works, the model should be validated by performing case studies at different organizations that pass the Pre Assessment of the CMM.

References

Abdelrazik. (2017). Docker vs. Kubernetes vs. Apache Mesos. Retrieved from https://mesosphere.com/blog/docker-vs-kubernetes-vs-apache-mesos/.

Alami, A. (2017). Defining Enterprise Agility. Retrieved from http://www.modernanalyst.com/Resources/Articles/tabid/115/ID/3864/Defining-Enterprise-Agility.aspx.

Amaral, M., Polo, J., Carrera, D., Mohomed, I., Unuvar, M., & Steinder, M. (2015, September). Performance evaluation of microservices architectures using containers. In *Network Computing and Applications (NCA), 2015 IEEE 14th International Symposium on* (pp. 27-34). IEEE.

Arnautov, S., Trach, B., Gregor, F., Knauth, T., Martin, A., Priebe, C., ... & Goltzsche, D. (2016, November). SCONE: Secure Linux Containers with Intel SGX. In *OSDI* (Vol. 16, pp. 689-703).

Ashrafi, N., Xu, P., Sathasivam, M., Kuilboer, J. P., Koelher, W., Heimann, D., & Waage, F. (2005, July). A framework for implementing business agility through knowledge management systems. In *E-Commerce Technology Workshops, 2005. Seventh IEEE International Conference on* (pp. 116-121). IEEE.

Banerjee, T. (2014, August 19). Understanding the key differences between LXC and Docker. Retrieved from https://archives.flockport.com/lxc-vs-docker/.

Bernstein, D. (2014). Containers and cloud: From lxc to docker to kubernetes. *IEEE Cloud Computing*, (3), 81-84.

Bessant, J., Knowles, D., Briffa, G., & Francis, D. (2002). Developing the agile enterprise. International Journal of Technology Management, 24(5-6), 484-497.

Bigelow, S. (n.d.). 5 Cons of container technology. Retrieved from https://searchservervirtualization.techtarget.com/feature/Five-cons-of-container-technology.

Bloom, N., & Pierri, N. (2018). Research: Cloud Computing Is Helping Smaller, Newer Firms Compete. *Harvard Business Review*.

Bloomberg, J. (n.d.) Think big with microservice: Lego-like software development takes shape. Retrieved from https://techbeacon.com/app-dev-testing/think-big-microservices-lego-software-development-takes-shape

Boer, A., & Van Engers, T. (2013). Legal knowledge and agility in public administration. *Intelligent Systems in Accounting, Finance and Management*, *20*(2), 67-88.

Booth, R. (1995). More agile than lean. In *Proceedings of the British Production and Inventory Control Society Conference* (pp. 191-207).

Breu, K., Hemingway, C. J., Strathern, M., & Bridger, D. (2002). Workforce agility: the new employee strategy for the knowledge economy. *Journal of Information Technology*, *17*(1), 21-31.

Brown, S., & Bessant, J. (2003). The manufacturing strategy-capabilities links in mass customisation and agile manufacturing–an exploratory study. International Journal of Operations & Production Management, 23(7), 707-730.

Buckley, K. (2017, January 10). Application containers will be a $2.7bn market by 2020, representing a small but high-growth segment of the Cloud-Enabling Technologies market. Retrieved from https://451research.com/blog/1351-application-containers-will-be-a-$2-7bn-market-by-2020,-representing-a-small-but-high-growth-segment-of-the-cloud-enabling-technologies-market.

Bui, T. (2015). Analysis of docker security. *arXiv preprint arXiv:1501.02967*.

Burgess, T. F. (1994). Making the leap to agility: defining and achieving agile manufacturing through business process redesign and business network redesign. *International Journal of Operations & Production Management*, *14*(11), 23-34.

Carroll, M., Kotzé, P., & Van Der Merwe, A. (2010). Going virtual: popular trend or real prospect for enterprise information systems.

Carroll, M., Kotzé, P., & Van der Merwe, A. (2011). Secure virtualization: benefits, risks and constraints.

Cho, H., Jung, M., & Kim, M. (1996). Enabling technologies of agile manufacturing and its related activities in Korea. *Computers & Industrial Engineering*, *30*(3), 323-334.

Christopher, M., & Towill, D. (2001). An integrated model for the design of agile supply chains. *International Journal of Physical Distribution & Logistics Management*, 31(4), 235-246.

Comer, D. E., & Stevens, D. L. (1993). Internetworking with TCP/IP: Client-Server Programming and Applications, volume III.

Connor, D. (2004). Server virtualization is on the rise. *Network World Canada*, *14*(23), 18.

Conroy, S. (2018, January 25). History of virtualization. Retrieved from https://www.idkrtm.com/history-of-virtualization/.

CoreOS. (n.d.). rkt. Retrieved from https://coreos.com/rkt/.

Dawson, P., & Bittman, T. J. (2008). Virtualization changes virtually everything. *Gartner Special Report*.

Dove, R. (1994). Tools for analyzing and constructing agility. In Proceedings of the Third Annual Agility Forum Conference/Workshop, Austin, TX.

Dove, R. (1999). Knowledge management, response ability, and the agile enterprise. *Journal of knowledge management*, *3*(1), 18-35.

Dove, R. (2001). Response ability. *The Language, Culture and Structure of the Agile Enterprise, New York*.

Dove, R. (2005a, March). Fundamental principles for agile systems engineering. In *Conference on Systems Engineering Research (CSER), Stevens Institute of Technology, Hoboken, NJ*.

Dove, R. (2005b, May). Agile enterprise cornerstones: knowledge, values, and response ability. In *IFIP International Working Conference on Business Agility and Information Technology Diffusion* (pp. 313-330). Springer, Boston, MA.

Dreyfus, S. E., & Dreyfus, H. L. (1980). A five-stage model of the mental activities involved in directed skill acquisition (No. ORC-80-2). California Univ Berkeley Operations Research Center.

Dreyfus, S. E. (2004). The five-stage model of adult skill acquisition. Bulletin of science, technology & society, 24(3), 177-181.

Dua, R., Raja, A. R., & Kakadia, D. (2014, March). Virtualization vs containerization to support paas. In Cloud Engineering (IC2E), 2014 IEEE International Conference on(pp. 610-614). IEEE.

Dyer, L., & Shafer, R. A. (2003). Dynamic organizations: Achieving marketplace and organizational agility with people. *CAHRS Working Paper Series*, 27.

Felter, W., Ferreira, A., Rajamony, R., & Rubio, J. (2015, March). An updated performance comparison of virtual machines and linux containers. In Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium On (pp. 171-172). IEEE.

Firesmith. (2017, September 25). Virtualization via Containers. Retrieved from https://insights.sei.cmu.edu/sei_blog/2017/09/virtualization-via-containers.html.

Fliedner, G., & Vokurka, R. J. (1997). Agility: competitive weapon of the 1990s and beyond?. *Production and Inventory Management Journal*, *38*(3), 19.

Flow.ci. (2016, September 29). Introduction to Containers: Concept, Pros and Cons, Orchestration, Docker, and Other Alternatives. Retrieved from https://medium.com/flow-ci/introduction-to-containers-concept-pros-and-cons-orchestration-docker-and-other-alternatives-9a2f1b61132c.

Forsythe, C. (1997). Human factors in agile manufacturing: a brief overview with emphasis on communications and information infrastructure. *Human Factors and Ergonomics in Manufacturing & Service Industries*, *7*(1), 3-10.

Fowler, M., & Foemmel, M. (2006). Continuous integration. (Thought-Works). Retrieved from https://www.martinfowler.com/articles/continuousIntegration.html.

Ganguly, A., Nilchiani, R., & Farr, J. V. (2009). Evaluating agility in corporate enterprises. *International Journal of Production Economics*, *118*(2), 410-423.

Ganore, P. (2014, December 5). Advantages and Disadvantages of Virtual Server. Retrieved from https://www.esds.co.in/kb/advantages-and-disadvantages-of-virtual-server/.

Gehani, R. R. (1995). Time-based management of technology: a taxonomic integration of tactical and strategic roles. *International Journal of Operations & Production Management*, *15*(2), 19-35.

Goldman, S. L., & Nagel, R. N. (1993). Management, technology and agility: the emergence of a new era in manufacturing. *International Journal of Technology Management*, *8*(1-2), 18-38.

Goldman, S. L., Nagel, R. N., & Preiss, K. (1995). *Agile competitors and virtual organizations: strategies for enriching the customer* (Vol. 8). New York: Van Nostrand Reinhold.

Gunasekaran, A. (1999). Agile manufacturing: a framework for research and development. *International journal of production economics*, *62*(1-2), 87-105.

Harauz, J., Kaufman, L. M., & Potter, B. (2009). Data security in the world of cloud computing" published by the IEEE computer and reliability societies in July.

Herzenberg, S. A., Alic, J. A., & Wial, H. (2000). *New rules for a new economy: Employment and opportunity in postindustrial America*. Cornell University Press.

Hevner, A., March, S., Park, J., & Ram, S. (2004). Design Science in Information Systems Research, *1*.

Higgins, J., Holmes, V., & Venters, C. (2015, July). Orchestrating docker containers in the HPC environment. In *International Conference on High Performance Computing* (pp. 506-513). Springer, Cham.

Hilton, P. D., Gill, G. K., & Little, A. D. (1994). CASE STUDY: ACHIEVING AGILITY: LESSONS FROM THE LEADERS. *Manufacturing Review*, *7*(2), 172-179.

Hohn, A. (2017). Introduction to Highly Available Container Applications. *DZone's Guide to Orchestrating and deploying Containers*. Retrieved from https://dzone.com/guides/orchestrating-and-deploying-containers.

Holbeche, L. (2015). The Agile Organization: How to build an innovative, sustainable and resilient business. Kogan Page Publishers.

Hopp, W. J., & OYEN, M. P. (2004). Agile workforce evaluation: a framework for cross-training and coordination. *Iie Transactions*, *36*(10), 919-940.

Humble, J., & Farley, D. (2011). Continuous delivery: reliable software releases through build, test, and deployment automation (pp. 115-117). Boston: Addison-Wesley.

Joy, A. M. (2015, March). Performance comparison between linux containers and virtual machines. In Computer Engineering and Applications (ICACEA), 2015 International Conference on Advances in (pp. 342-346). IEEE.

Kaddoumi, T., & Watfa, M. (2016, August). A proposed agile enterprise architecture framework. In *Innovative Computing Technology (INTECH), 2016 Sixth International Conference on* (pp. 52-57). IEEE.

Kaur. (2018). What is Docker?. Retrieved from https://dzone.com/articles/what-is-docker-1.

Kidd, P. T. (1994). Agile Manufacturing Forging New Frontiers. Addisson-Wesley. *Reading*.

Kidd, P. T. (1996). Agile manufacturing: a strategy for the 21st century.

Korn Ferry. (2014, April 16). Retrieved from https://www.kornferry.com/enterprise-agility.

Kumar, A., & Motwani, J. (1995). A methodology for assessing time-based competitive advantage of manufacturing firms. *International Journal of Operations & Production Management*, *15*(2), 36-53.

Lee, J., Siau, K., & Hong, S. (2003). Enterprise Integration with ERP and EAI. Communications of the ACM, 46(2), 54-60.

Lewis, J., & Fowler, M. (2014). Microservices: a definition of this new architectural term. Mars.

Li, W., & Kanso, A. (2015, March). Comparing containers versus virtual machines for achieving high availability. In *Cloud Engineering (IC2E), 2015 IEEE International Conference on* (pp. 353-358). IEEE.

Ligus, R. G. (1993). Enterprise agility: Maneuverability and turbo power. INDUSTRIAL MANAGEMENT-CHICAGO THEN ATLANTA-, 35, 27-27.

Luo, X., Yang, L., Ma, L., Chu, S., & Dai, H. (2011). Virtualization security risks and solutions of Cloud Computing via divide-conquer strategy. In *Multimedia Information Networking and Security (MINES), 2011 Third International Conference on* (pp. 637-641). IEEE.

Mathiyalakan, S., Ashrafi, N., Zhang, W., Waage, F., Kuilboer, J.P., and Heimann, D. "Defining Business Agility: An Exploratory Study," Forthcoming in the Proceedings of the 16th Information Resource Management Association International Conference, San Diego, CA, May 15-18, 2005.

Mccarty, F.H., 1993. Agility in Manufacturing. Manufacturing Engineering 111 (6), 8.

McCoy, D. W., & Plummer, D. C. (2006). Defining, cultivating and measuring enterprise agility. *Gartner research*, *28*.

McKinsey&Company. (n.d.). Enterprise Agility. Retrieved from https://www.mckinsey.com/business-functions/organization/how-we-help-clients/enterprise-agility.

Menor, L. J., Roth, A. V., & Mason, C. H. (2001). Agility in retail banking: a numerical taxonomy of strategic service groups. *Manufacturing & Service Operations Management*, *3*(4), 273-292.

Morabito, R., Kjällman, J., & Komu, M. (2015, March). Hypervisors vs. lightweight virtualization: a performance comparison. In *Cloud Engineering (IC2E), 2015 IEEE International Conference on* (pp. 386-393). IEEE.

Morris, K. (2016). Infrastructure as code: managing servers in the cloud. " O'Reilly Media, Inc.".

Nagel, R. N., & Dove, R. (1991). *21st century manufacturing enterprise strategy: An industry-led view*. Diane Publishing.

Nejatian, M., & Zarei, M. H. (2013). Moving towards organizational agility: Are we improving in the right direction?. *Global Journal of Flexible Systems Management*, *14*(4), 241-253.

NH Learning Solutions. (2017, February 13). Top 5 Ways Businesses Benefit from Server Virtualization. Retrieved from https://blog.nhlearningsolutions.com/blog/top-5-ways-businesses-benefit-from-server-virtualization.

Oliver, T., Muir, M. (2017, June 22). Defining Enterprise Agility. Retrieved from http://blog.deloitte.com.au/defining-enterprise-agility/.

Oracle. (2012). *Oracle® VM*. 1.1.1. Brief History of Virtualization. Retrieved from https://docs.oracle.com/cd/E26996_01/E18549/html/VMUSG1010.html

Oracle. (2014, December). Zones Overview. Retrieved from https://docs.oracle.com/cd/E36784_01/html/E36848/zones.intro-2.html#scrolltoc.

Osnat, R. (2018, March 21). A Brief History of Containers: From the 1970s to 2017. Retrieved from: https://blog.aquasec.com/a-brief-history-of-containers-from-1970s-chroot-to-docker-2016.

Overby, E., Bharadwaj, A. & Sambamurthy, V. Eur J Inf Syst (2006) *Enterprise Agility and the enabling role of information technology*. 15: 120. https://doi.org/10.1057/palgrave.ejis.3000600.

Pahl, C. (2015). Containerization and the paas cloud. *IEEE Cloud Computing*, *2*(3), 24-31.

Pahl, C., Brogi, A., Soldani, J., & Jamshidi, P. (2017). Cloud container technologies: a state-of-the-art review. *IEEE Transactions on Cloud Computing*.

Pahl, C., & Lee, B. (2015, August). Containers and clusters for edge cloud architectures--A technology review. In *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on* (pp. 379-386). IEEE.

Pal, N., & Pantaleo, D. (Eds.). (2005). The agile enterprise: Reinventing your organization for success in an on-demand world. Springer Science & Business Media.

Paulk, M. C., Curtis, B., Chrissis, M. B., & Weber, C. V. (1993). The capability maturity model for software. *Software engineering project management*, *10*, 1-26.

Perrochon, L. (1994, November). Translation servers: Gateways between stateless and stateful information systems. In *Network Services Conference*.

Pietroforte, M. (2008, July 3). Seven disadvantages of server virtualization. Retrieved from https://4sysops.com/archives/seven-disadvantages-of-server-virtualization/.

Pinochet, A., Matsubara, Y., & Nagamachi, M. (1996). Construction of a knowledge-based system for diagnosing the sociotechnical integration in advanced manufacturing technologies. *International Journal of Human Factors in Manufacturing*, *6*(4), 323-348.

Plonka, Francis E. "Developing a lean and agile work force." *Human Factors and Ergonomics in Manufacturing & Service Industries* 7, no. 1 (1997): 11-20.

Porter, A. L. (1993). Virtual companies reconsidered: Essay review. Technology analysis & strategic management, 5(4), 413-420.

Portworx. (2017). Portworx Annual Container Adoption Survey 2017.

Prahalad, C. K., & Hamel, G. (1999). The core competence of the corporation. In *Knowledge and strategy* (pp. 41-59).

PricewaterhouseCoopers LLP. (2015). *Building Enterprise Agility*. PwC Technology Institute.

PricewaterhouseCoopers (n.d.). History and milestones. Retrieved from https://www.pwc.com/us/en/about-us/pwc-corporate-history.html.

PricewaterhouseCoopers NL (n.d.). PwC in Nederland. Retrieved from https://www.pwc.nl/nl/onze-organisatie/pwc-in-nederland.html.

Project Management Institute. (2008). Organizational Project Management Maturity Model (OPM3®) (2nd ed.). Newtown Square, PA: Project Management Institute.

Raschke, R. L., & David, J. S. (2005). Business process agility. *AMCIS 2005 Proceedings*, 180.

Reed, K., & Blunsdon, B. (1998). Organizational flexibility in Australia. *International Journal of Human Resource Management*, *9*(3), 457-477.

Ren, J., Yusuf, Y. Y., & Burns, N. D. (2000, March). A prototype of measurement system for agile enterprise. In The Third International Conference of Quality Reliability Maintenance (pp. 29-30).

Ren, J., Yusuf, Y. Y., & Burns, N. D. (2003). The effects of agile attributes on competitive priorities: a neural network approach. *Integrated Manufacturing Systems*, *14*(6), 489-497.

Rigby, D. K., Sutherland, J., & Takeuchi, H. (2016). Embracing agile. Harvard Business Review, 94(5), 40-50.

Roberts, N., & Grover, V. (2012). Investigating firm's customer agility and firm performance: The importance of aligning sense and respond capabilities. *Journal of Business Research*, *65*(5), 579-585.

Sambamurthy, V., Bharadwaj, A., & Grover, V. (2003). Shaping agility through digital options: Reconceptualizing the role of information technology in contemporary firms. *MIS quarterly*, 237-263.

Sampathkumar, A. (2013). Virtualizing Intelligent River®: A Comparative Study of Alternative Virtualization Technologies.

Scroggins, R. (2013). Virtualization technology literature review. *Global Journal of Computer Science and Technology*.

Scheepers, M. J. (2014, June). Virtualization and containerization of application infrastructure: A comparison. In 21st Twente Student Conference on IT (Vol. 1, No. 1, pp. 1-7).

Seo, K. T., Hwang, H. S., Moon, I. Y., Kwon, O. Y., & Kim, B. J. (2014). Performance comparison analysis of linux container and virtual machine for building cloud. Advanced Science and Technology Letters, 66(105-111), 2.

Sharifi, H., & Zhang, Z. (1999). A methodology for achieving agility in manufacturing organisations: An introduction. International journal of production economics, 62(1), 7-22.

Sharma, P., Chaufournier, L., Shenoy, P., & Tay, Y. C. (2016, November). Containers and virtual machines at scale: A comparative study. In Proceedings of the 17th International Middleware Conference (p. 1). ACM.

Sharp, J. M., Irani, Z., & Desai, S. (1999). Working towards agile manufacturing in the UK industry. International Journal of production economics, 62(1-2), 155-169.

Sherehiy, B., Karwowski, W., & Layer, J. K. (2007). A review of enterprise agility: Concepts, frameworks, and attributes. International Journal of industrial ergonomics, 37(5), 445-460.Dawson, P., & Bittman, T. J. (2008). Virtualization changes virtually everything. *Gartner Special Report*.

Showell, J. (2015, November 23). 8 pros and cons of containers for app deployments. Retrieved from http://www.serverspace.co.uk/blog/8-pros-and-5-cons-of-containers-for-app-deployments.

Spiegel, E. (2006). Real World Virtualization: Realizing the Business Benefits of Application and Server Virtualization. *Computer Technology Review*, *26*(2), 8-8.

Sysprobs. (n.d.). Disadvantages virtualization, What's your Opinion?. Retrieved from http://www.sysprobs.com/disadvantages-virtualization-opinion.

Tamane, S. (2015). A review on virtualization: A cloud technology. *International Journal on Recent and Innovation Trends in Computing and Communication*, *3*(7), 4582-4585.

Techopedia. (n.d.). Virtualization. Retrieved from https://www.techopedia.com/definition/719/virtualization.

Team, C. P. (2002). Capability maturity model® integration (CMMI SM), version 1.1. CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI-SE/SW/IPPD/SS, V1. 1).

Tesfatsion, S. K., Klein, C., & Tordsson, J. (2018, March). Virtualization Techniques Compared: Performance, Resource, and Power Usage Overheads in Clouds. In Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering (pp. 145-156). ACM.

The Kubernetes Authors. (n.d.). Production-Grade Container Orchestration. Retrieved from https://kubernetes.io/.

The Open Group. (2016). *Archimate® 3.0 specification*. Retrieved from https://publications.opengroup.org/c162.

Thrive Operations. (2018, January 11). Benefits of Virtualization. Retrieved from https://www.thrivenetworks.com/blog/benefits-of-virtualization/.

Tosatto, A., Ruiu, P., & Attanasio, A. (2015, July). Container-based orchestration in cloud: state of the art and challenges. In *Complex, Intelligent, and Software Intensive Systems (CISIS), 2015 Ninth International Conference on* (pp. 70-75). IEEE.

Tsourveloudis, N. C., & Valavanis, K. P. (2002). On the measurement of enterprise agility. *Journal of Intelligent and Robotic Systems*, *33*(3), 329-342.

Tseng, Y.-H., & Lin, C.-T. (2011). *Enhancing enterprise agility by deploying agile drivers, capabilities and providers*. Information Sciences, *181*(17), 3693–3708. http://doi.org/https://doi.org/10.1016/j.ins.2011.04.034

Van de Weerd, I., & Brinkkemper, S. (2009). Meta-modeling for situational analysis and design methods. In *Handbook of research on modern systems analysis and design technologies and applications* (pp. 35-54). IGI Global.

Van Herpen, D. (2014). *Four pattern for enterprise agility*. Retrieved from https://www.sogeti.nl/updates/blog/four-patterns-enterprise-agility

Van Oyen, M. P., Gel, E. G., & Hopp, W. J. (2001). Performance opportunity for workforce agility in collaborative and noncollaborative work systems. *Iie Transactions*, *33*(9), 761-777.

Vaquero, L. M., Rodero-Merino, L., Caceres, J., & Lindner, M. (2008). A break in the clouds: towards a cloud definition. ACM SIGCOMM Computer Communication Review, 39(1), 50-55.

Vastag, G., Kasarda, J. D., & Boone, T. (1994). Logistical support for manufacturing agility in global markets. *International Journal of Operations & Production Management*, *14*(11), 73-85.

Waters, J. K. (2007, March 15). Virtualization Definition and Solutions. Retrieved from https://www.cio.com/article/2439494/virtualization/virtualization-definition-and-solutions.html.

Wieringa, R.J. (2014) *Design Science Methodology for Information Systems and Software Engineering*. Springer Verlag

Xavier, M. G., Neves, M. V., Rossi, F. D., Ferreto, T. C., Lange, T., & De Rose, C. A. (2013, February). Performance evaluation of container-based virtualization for high performance computing environments. In Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on (pp. 233-240). IEEE.

Yigang, W. H. W. C. W. (1999). Evaluating System of Enterprise Agility [J]. *Systems Engineering*, *6*, 006.

Youndt, M. A., Snell, S. A., Dean Jr, J. W., & Lepak, D. P. (1996). Human resource management, manufacturing strategy, and firm performance. *Academy of management Journal*, *39*(4), 836-866.

Youssef, M. A. (1992). Agile manufacturing: a necessary condition for competing in global markets. *INDUSTRIAL ENGINEERING-NEW YORK THEN ATLANTA NORCROSS-AMERICAN INSTITUTE OF INDUSTRIAL ENGINEERS INCORPORATED-*, *24*, 18-18.

Youssef, M. A. (1994). Design for manufacturability and time-to-market, Part 1: theoretical foundations. *International Journal of Operations & Production Management*, *14*(12), 6-21.

Yusuf, Y. Y. (1996). The extension of MRPII in support of Integrated Manufacture. *Unpublished Ph. D. Thesis, University of Liverpool*.

Yusuf, Y., Sarhadi, M., Gunasekaran, A., 1999. Agile manufacturing: the drivers, concepts and attributes. International Journal of Production Economics 62 (1–2), 33–43

Appendix I – Agility: definitions

#	Authors and year	Definition
1	Nagel and Dove, 1991	A system that shifts quickly among product models/lines, ideally in real time in order to respond to customer needs.
2	Kidd, 1994	A rapid and proactive adaption of enterprise elements to unexpected and unpredicted changes, and represents a new and radically different manufacturing business model.
3	Goldman et al., 1995	Capability of an organization to operate profitably in a competitive environment comprised of continually changing customer habits.
4	Kumar and Motwani, 1995	Ability to accelerate the activities on critical path and time-based competitiveness.
5	Cho, Jung and Kim, 1996	Capability to survive and prosper in a competitive environment or continuous and unpredictable changes by reacting quickly and effectively to changing markets, designed by customer designed products and services.
6	Fliedner and Vokurka, 1997	Ability to market successfully low-cost, high- quality products with short lead times and in varying volumes that provide enhanced value to customers through customization.
7	Yusuf et al., 1999	Agility is the successful exploration of competitive bases (speed, flexibility, innovation proactivity, quality, and profitability) through the integration of reconfigurable resources and best practices in a knowledge-rich environment to provide customer-driven products and services in a fast changing market environment.
8	Sharifi and Zhang, 1999	Agility is the ability to detect the changes in the business environment, and respond to them by providing the appropriate capabilities.
9	Dove, 1999 2x Dove, 2001 1x Dove, 2005b 1x	The ability of an organization to thrive in a continuously changing, unpredictable business environment. & Agility is the ability to manage and apply knowledge effectively. Ability of an organization to respond efficiently and effectively to both proactive and reactive needs and opportunities on the face of an unpredictable and uncertain environment (2001)*. *This definition is not a given definition, but a combination of characteristics that are provided in the book of Dove (2001), and processed into a definition by Ganguly et al. (2009). Later on, Dove (2005b) does provide a definition that incorporates these characteristics: "*Agile systems – for both reactive and proactive response needs and opportunities – when these are unpredictable, uncertain, and likely to change*". This definition is given in this row to increase clarity of the topic regarding this author.
10	Menor, Roth, and Mason, 2001	The ability of a firm to excel simultaneously on operations capabilities of quality, delivery, flexibility and cost in a coordinated fashion.
11	Bessant, Knowles, and Briffa, 2002	The organization's capacity to gain competitive advantage by intelligently, rapidly, and proactively seizing opportunities and reacting to threats.
12	Brown and Bessant (2003)	The ability to respond quickly and effectively to current market demands, as well as being proactive in developing future market opportunities.
13	Sambamurthy et al. (2003)	Ability of a firm to redesign their existing processes rapidly and create new processes in a timely fashion in order to be able to take advantage and thrive of the unpredictable and highly dynamic market conditions.
14	Ashrafi et al. (2005)	An organization's ability to sense environmental changes and respond effectively and efficiently to that change.
15	Raschke and David (2005)	Ability of a firm to dynamically modify and/ or reconfigure individual business processes to accommodate required and potential needs of the firm'.
16	Mathiyakalan et al. (2005)	Ability of an organization to detect changes (which can be opportunities or threats or a combination of both) in its business environment and hence providing focused and rapid responses to its customers and stakeholders by reconfiguring its resources, processes and strategies

Appendix II – Retrospective view on Enterprise Agility

Analysis of scientific sources show that "enterprise agility" was already mentioned during the first decade (1991-1999). For the analysis, the key term "enterprise agility" was used to search into Google Scholar to find the study that mentions EAG for the first time. As stated before, it occurs that enterprise agility is meant as agility for an enterprise, with the exact meaning of agility during the first decade (reactive and enterprise component-specific). Therefore, multiple sources are given in this brief analysis to depict the difference. To increase the semantical distinction, "*enterprise agility*" is written in this text when conventional agility is meant. "EAG" is stated when the specific, self-contained concept of EAG (proactive and enterprise-wide) is meant.

The earliest mention of EAG was in 1993 in the article of '*Virtual Companies Reconsidered*' by Porter (1993). Here, Porter (1993) criticized the concept of virtual companies regarding the required policies of each organization, and denoted the issue of how to accommodate enterprise agility with the organization's security. In this paper, *enterprise agility* is meant as the overall agility of an organization, and not as the specific, self-contained concept EAG as in the second decade.
At the ending of the same year, Ligus (1993) wrote an introducing paper about agility: '*Enterprise agility: Maneuverability and turbo power*'. The author argued about the advantages of an organization being agile, translating it into the term enterprise agility. However, this paper aimed at the overall agility of the organization, and not the specific concept of EAG.

The subsequent year, prominent agility researcher Dove (1994) discussed analytical tools for constructing and evaluating agile strategies. 'Enterprise agility' is mentioned three times in the article. The first time, *enterprise agility* is meant anew as agility of an enterprise. The second time refers to the object of interest provided in this paper: 'Structure of Enterprise Agility' (third time). This structure consists of agile attributes, change domains, enterprise elements and agile dimensions, and forms the foundation of the work of several prominent agility researchers as mentioned before in Section 5.1.1.1 (Sharifi & Zhang, 1999; Yusuf et al., 1999; Tseng & Lin, 2011). Interesting fact is that the structure depicts possible application of agile concepts in different enterprise elements, which corresponds with the EAG's meaning of extending agility throughout the whole organization. However, ten out of 12 of the denoted enterprise elements are specifically for the field of manufacturing, and not for remaining departments of organizations. Therefore, despite its relevance for the development of agility, the usage of *enterprise agility* in this article is not in line with EAG.

Agility for manufacturing was 'invented' because the USA experienced competitive pressure from Europe and Asia. During the first decade, Asia was also investigating agility. In 1999, Yigang (1999) developed an evaluation system to evaluate agility of enterprises, based on the work of Dove (1994). Similary to the aforementioned article, the term *enterprise agility* was used since the artefact is a system that evaluates an enterprise's agility, what was mainly aimed at AM during that period, and is therefore not lined with EAG.

At last, in the book of Dove (2001) about organizations' response ability and agile enterprises, enterprise agility is mentioned two times. Despite the fact that *enterprise agility* is not explicitly defined, and is again used to denote the agility of an organization (the enterprise), the book describes more than solely agility techniques for enterprise manufacturing components. It provides descriptions about agility and its surroundings, change-enabling structure and culture, adaptable structures, response-able systems and -enterprises, and change proficiency. The latter is the aforementioned required distinction between reactive and proactive demeanor. These are the reasons that these descriptions are in line with the concept of EAG, as they are focused on the whole enterprise becoming agile. The second reason is the explanation of change proficiency, realizing the introduction of the proactive demeanor, what eventually evolves into the sensing ability.

The results of this brief analysis show the first use of the term 'enterprise agility' in literature, whereas agility of an enterprise is meant with corresponding reactive demeanor, partial agility and main focus on the field of manufacturing. 'Enterprise agility' in terms of EAG, with corresponding proactive demeanor, agility throughout the whole organization, and the ability of sensing, is not meant. However, Dove (2001) does refer to EAG in its description of the agile enterprise. Although this is not explicitly, it does corresponds with found EAG components.

Appendix III – Virtualization types

Native virtualization
Software is launched to mimic a machine. This VM uses the same hardware as the underlying system. Hence, an x86 platform can only virtualize x86 VMs, and is unable to virtualize x64 platforms. Furthermore, the software that is being loaded to realize the VMs (the Host OS) divides the available resources among its virtual machines. This concept realizes a better utilization of resources from the host by the VMs (guests Oss). Specific examples are VMware Server, -Workstation, -Player, Microsoft virtual PC/Server, Vserver and Qemu.

Full virtualization
Full virtualization enables multiple VMs being loaded next to each other on a set of (physical) hardware. This is realized by adding a software layer between the hardware and virtual machines. This layer manages tasks and requests that come from the virtual machines towards the hardware. This can be seen as placing a virtual host on the physical hardware to translate virtual actions as performed by VMs. An example of such virtualization is VMware ESX server.

Partial virtualization
Virtual components are made of a selection of hardware components, instead of virtualization of a full set of hardware components that mimic a computer. These components or instances realize that it is possible to share devices. However, it is not possible to share more than one OS. Although this type of virtualization is not recognized as virtualization, it is widely used in OSs as Windows and Linux. Partial virtualization is also applicable on large mainframe systems.

Paravirtualization
Hardware is offered to a virtual machine through specialized APIs (Application Programming Interface) that have to conform to the Guest OS. By using this technique, virtual machines can determine what hardware is going to be shared with other virtual machines, and which hardware is specifically designated for a certain virtual machine. This technique is applied by Xen, Trango, and Sun logical domains.

Application virtualization
In this type of virtualization, an application runs while using the local system resources (desktop or laptop) inside a modified virtual machine, without being installed on that particular system. Benefit is that application updates can be installed without being dependent on the local system the application runs on. A benefit is that it is possible to execute applications with conflicting requirements together on the same system. Examples are Thinstal, Microsoft Application virtualization, Altiris SVS, Sun Java VM, and Trigence.

Server virtualization
Server virtualization is the masking of server resources from server users. This includes the number and identity of individual physical servers, CPUs and operation systems. The goal is to spare the user the complex configuration of server resources, while increasing resource sharing and utilization. Additionally, the capacity is maintained to have expanding possibilities when needed.

Resource virtualization
Most used is the virtualization of *storage* and *networking* elements. **Storage virtualization** enables the pooling of physical storage from multiple network devices. By doing so, it appears to be a single storage device that is managed by a centralized console. **Network virtualization** combines available resources in a network by splitting up available bandwidth into channels. Examples of networking elements are VLANs and VPNs. Another type is the virtualization of *memory* (RAM). Here, aggregating RAM resources from networked systems into a single memory pool.

Data virtualization
The presentation of data as an abstract layer, independent of underlying database systems, structures and storage. Possibly combined with **Database virtualization**, what is the decoupling of the database layer, which lies between storage and application layers within the application stack.

Appendix IV – Containerization vendors

Container technologies

The container domain consists of two types of technologies: 1) container solutions and 2) container orchestrators (see subsection 2.3.3). This appendix elaborates on the different types of container solutions. These container solutions provide the same type of container services (deploy, manage, and scale applications), but differentiate from each other when going into details. In reality, each solution can be used to solve different things and are rooted in varying contexts (Abdelrazik, 2017). This is due to their individual specializations, applied techniques, and their own goals.

The most popular container solution is Docker. According to aforementioned survey of Portworx (2017), 79% of the organizations use Docker as their primary container technology. Docker provides a container engine on top of the Host OS, which manages all containers on that collection of hardware. It does that by combining Linux containers with an overlay filesystem and by providing tools for building and packaging applications into portable environments, i.e., container images (Morabito, Kjällman, Komu, 2015). Primarily, the solution was Linux-based – creating Linux containers – but nowadays Docker also provides Windows-based containers. Docker differentiates itself because it is a single application virtualization engine based on containers (Banerjee, 2014). Meaning that each Docker container only facilitates one application. This results in loosely coupled 'frozen' applications as services, which enables users to convert these frozen states of applications to other machines and OSs.

- *rkt*: Linux-based application container engine for modern production cloud-native environments. Its core execution unit is the pod, what is a collection of one or more applications in multiple containers executing in a shared context (similar to Docker) (CoreOS, n.d.).
- *FreeBSD Jails*: works with Linux userspace isolation as additional security. This means that process only communicates with other processes in the same container.
- *LXC*: a container technology which enables lightweight Linux containers by providing base OS container templates and a comprehensive set of tools for container lifecycle management. Users can approach these LXC containers as an OS and install applications and services on it, while expecting the correct support (Banerjee, 2014).
- *Solaris Zones*: virtualizes operating system services and provides an isolated and secure environment for executing applications. A zone is a virtualized operating system environment created within a single instance of the Oracle Solaris OS (a global zone) (Oracle, 2014).

Container orchestrators

As shown by Portworx (2017), organizations use multiple orchestration tools. This is because vendors are continuously in development and keep providing new functionalities. Another reason is the prevention of vendor lock in. This shows that currently the orchestration tooling market is dynamic.

This dynamic market consists of multiple vendors that provide container orchestration tooling by using their specific techniques. Examples of popular vendors are: Docker Swarm, Kubernetes, Mesos, Cloud Foundry, Amazon ECS, and Azure Container Service. Kubernetes is the most used orchestration tool in organizations, as 43% of participating organizations indicate that Kubernetes is applied as container orchestrator (Portworx, 2017). Moreover, 32% states that Kubernetes is their primary orchestration tool. Especially larger organizations (5000+ employees), are using Kubernetes (respectively 48% and 33%).

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications (The Kubernetes Authors, n.d.). Kubernetes is based on processes that run on Docker hosts. The orchestrator binds hosts into clusters, and manages containers by grouping them into logical units (Pahl, 2015b; The Kubernetes Authors, n.d.). It schedules computing resources to containers and the other way around, which enables rapid scaling (Tosatto et al., 2015).

Appendix V – PI: Interview protocol

Goal: to find out how the container technology can support organizations in their enterprise agility.

Introduction

- Introducing myself;
- Tell about the master;
- Explain the master thesis project, including structure, deliverables, expectations;
- Explain the goal, structure, subjects, and timespan of this interview, and why this person;
- Ask for permission to record this interview. The record will only be used to make a transcription and additional notes, in order to improve the quality of the research.

Start recording with phone

General

"**I would like to start with some general information.**"

1. Ask interviewee to tell something about him-/herself, position, daily tasks etc.;
2. What do you think about the role of enterprise agility (EAG) in organizations nowadays?
3. EAG is defined as the following: **Show paper with definition**.
   To what extend do you agree with this?
4. What is agile according to you? And what would be an agile aspect according to you?

Enterprise agility

"Both scientists and practitioners extensively study this concept. During the literature review, I found that most important enterprise components are Organization, People, and Technology (OPT). As the literature review showed, research is conducted for O & P towards agility. This resulted in different agility types specifically for O & P domains. However, no research (lines) dedicated to Technology exists in the literature. The only made scientific statements denote the importance of technology for organizations, but no (tangible) recommendations to apply a certain technology to improve EAG. Hence, there is a lack of relationship between EAG and the potential benefit of technology".

5. To what extent do you think this statement applies in your field or work?
6. Why (not)? Could you elaborate on this?
7. **IF APPLIES**: How do you think this gap can be filled?
8. **IF NOT APPLIES**: What would be a current explicit link between technologies and EAG?
   a. Can these technologies be seen as agile technologies, or become a part of "technology agility"?
9. What are in your opinion agile technologies, or 'technologies that contain agile aspects', or 'technologies that enable the agile Way of Working' (WoW)?
10. What would be an agile (enabling) technology from your perspective?

Containerization

"Virtualization enables organizations to manage their hardware more efficiently, and increase utilization. Of course cloud is enabled by the technology. However, since 2013 containers became popular."

11. How are containers currently improving organizations?
12. What are in your opinion the benefits of containers? What popularized them?
13. Do you consider containers as a technology that facilitate agile aspects or an agile environment? Or enables these? Why (not)?

"According to what we just talked about concerning EAG and technology:"

14. How can containers support organizations in their enterprise agility in your opinion?
    a. How would you define this? Let's say in what kind of entities (e.g. guidelines, principles, rules, characteristics, process, etc)?

"During the meetings with my supervisor of the university, we discussed about a potential artefact resulting from this project. The first concept of this artefact is an integration process for containers to deploy them in the (enterprise) architecture, to ultimately increase the EAG of an organization."

    b. Do you think it is possible to make a process-based solution for implementing containers with respect to EAG?

Show the results of the discussion with the first interviewee --> containers applied in a part of software development process, including capabilities

    c. Do you think the field of practitioners can benefit from such an artefact?
    d. What would you change about it?

"Since the increasing amount of containers in organizations, and the importance of orchestrators, the role of containers is growing."

15. How do you think containers, or large amounts of container clusters, can be more efficiently used? Thus, increasing EAG?
16. **Therefore,** can containers be a technology that can be explicitly linked to EAG as an EAG enabling technology?
17. How would you see the application of containerization in five years?

Finalizing

18. Is there something I forgot to ask about, what you consider as important?
19. Do you want to add something else?

*End of interview

- Thank interviewee for the interview;
- Offer to send the results when project is finished;
- Ask for possibility of additional questions if later required;
- Ask if interviewee wants to see the transcript to check validity.

Appendix VI – PI: Expert interview results

The interviewees (experts) have been partly anonymized due to made confidentiality agreements. Their input and quotes are removed from the public version of the thesis.

Results interview 1 – Expert 1

Results interview 2 – Expert 2

Results interview 3 – Expert 3

Results interview 4 – Expert 4

Appendix VII – PI: Expert interview results container areas

The interviewees (experts) have been partly anonymized due to made confidentiality agreements. Their input and quotes are removed from the public version of the thesis.

Results interview 1 – Expert 1

Results interview 2 – Expert 2

Results interview 3 – Expert 3

Results interview 4 – Expert 4

Appendix VIII – Virtualization technologies linked to EAG

This appendix elaborates on the core agility attributes that are perceived as EAG characteristics. Table 30 explains the attributes by using the quoted descriptions of Sherehiy et al. (2007). We additionally provided the corresponding sub-concepts that are relevant only to technology in the table, and gave an example for this research's context. Finally, for each attribute, we rationalized how we linked them to virtualization concepts.

Table 30: Core agility attributes including technology-related sub-concepts (Sherehiy et al., 2007) linked to VMs and containers

Concept	Description
Flexibility	"*The ability to pursue different business strategies and tactics, to quickly change from one strategy/**task**/job to another.*" - Flexible organizational structures and practices (i.e. capacity of applications and infrastructure can be scaled or changed in balance depending on demands); - Workplace flexibility (i.e. all kinds of applications can be launched through virtual environments, independently of the location and infra agnostic).
	Link to VMs: VMs provide a virtual environment where applications run into to execute their services. Because of the VMs, physical hardware can be virtually divided into multiple virtual hardware instances. This enables an organization to run more different OSs and applications on their hardware, and the ability to scale up or down dependent on the users' demands.
	Link to containers: containers provide equivalent services as VMs, however faster and through different functionality. Containers are infra agnostic, which means that they can independently run on any kind of hardware, making organizations more agile in providing infrastructure for virtual environment purposes. Besides that, containers use lightweight OS images and solely include the required runtime packages to execute the desired application. This makes them resource efficient, quickly to launch, and customized for specific applications. Ultimately, lending themselves for microservices architectures, as individual services can be launched in containers, based on and easily scaled to user demands. Moreover, if business logic is positioned into containers, then the stateless characteristic of containers can be used. When launched, the container receives input, executes business logic, and provides output (input'). After the output delivery, the container is immediately destroyed and all data within the container is vanished. This stateless aspect makes containers flexible as well, since data does not have to be stored, computing resources do not have to be kept in reservation, and containers only run whenever they are needed to facilitate an application instance.
Speed	"*The ability to complete requirements of all other agile characteristics in shortest possible time. The ability to learn, carry out tasks and operations and make changes in shortest possible time.*" - Performing tasks, operations, and making changes (i.e. fastness in launching instances of applications and infrastructure (application run time environments) to perform tasks and operations); - Time of: operations, production/service delivery (i.e. fastness of an application is able to start running a service).
	Link to VMs: the same as with Flexibility, VMs provide a virtual environment for OSs and applications. As a result, organizations were not required to launch additional physical servers whenever they needed different types of OSs anymore. This increased the process speed of launching another application, or another OS and application together. Moreover, process can also be automated, further increasing the deployment of hardware.
	Link to containers: as aforementioned for Flexibility, containers provide equivalent services, although launched and decommissioned faster (lightweight OS and rationalized packages). This process can also be automated, further enhancing its speed. Moreover, whereas VMs require multiple configuration files to be launched onto the correct infrastructure and with the correct settings, containers are launched and configured by the means of basic code commands or config-files.
Responsiveness	"*The ability to identify changes and opportunities and respond reactively or proactively to them.*" - Responsiveness to change in customer demands (i.e. scalability for application capacity);

	- Responsiveness to market and business environment changes and trends (i.e. rapidly integrating or executing new applications in organization's application and infrastructure landscape, in order to respond to or anticipate on market needs).
	Link to VMs: virtual environments of VMs, which contain application instances, can be duplicated to meet capacity requirements based on user demands. This means that applications/services capacity are increased so that increasing amounts of user can access their desired application/service. If organizations have automated this process, capacity is scaled automatically based on the received demands, further improving the responsiveness of organizations.
	Link to containers: containers can also be duplicated to increase capacity of an application/service. However, containers can increase the capacity more accurately, as an individual service can be put into a container (synergy with MSA pattern). Whereas in VMs, it would be inefficient to take the effort to launch a VM with full OS image to increase the capacity of one service. Primarily, because of this extensive configuration and full OS image, complete application duplicates are launched within VMs (no synergy with MSA pattern), removing the fast responsiveness for scalability and immediate deployments.
Integration and Low complexity	*"Close and simple relations between the individual system components, easy and effortless flow of the materials, information and communication between the system components, organizational structures, people, and technology."* - Integration of people, technology, and organization (i.e. integration of DevOps runtime environments of applications by the means of containers, while using the IaC principle); - Synthesis of diverse technologies, skills, and competencies (i.e. combination of microservices architecture with containers and IaC); - Low complexity of structure, relationships between structure elements (i.e. technical architecture, applications/services that are independently managed with determined runtime environment); - Flow of material, communication and information between different organizational structures and systems components (i.e. networking maturity, process maturity, API configuration); - Enhanced interaction between processes.
	Link to VMs: standardized OS and application images can be used for VMs to launch predefined instances of environments and applications. The degree of standardized, organization-wide obligated images and predefined connections with the internal network reduces complexity of the application and infrastructure landscape.
	Link to containers: the rationale about images also applies to the leaner container images. In addition, the easily configured scripts and code commands for launching containers also reduce complexity. The stateless aspect supports low complexity even more. If a service in a container already withholds business logic, the incoming information is modified and directly returned back or forwarded to the next user or service. Preferably, by the means of an (standardized) API. This means that the container is simply destroyed when it is no longer required, and the modified data does not have to be maintained in the container. If required, data is stored at a component that is specialized for this task.
Culture of change	*"A description of environment supportive of experimentation, learning, and innovation and is focused on the continuous monitoring environment to identify changes. Culture of change is an environment where people on all organizational levels have positive and fearless attitude to changes, different opinions, new ideas, and technology."* We did not incorporate this core attribute into the relationship to both containers and VMs, as the CMM. The reason is that this attribute is focused on people (O of OPT), and is therefore according to us not directly related to our aim of technology (T of OPT).

Appendix IX – Pre assessment rationale elaboration

This appendix describes the rationale for each question of the Pre assessment.

Table 31: Pre assessment rationale per question

Pre. Area	#	Rationale per question
General	1	This is an important question for the Maturity assessment. If an organization does do anything related to software developing, the CMM is not completely relevant to apply for that organization.
	2	The outsourcing aspect provides us with an indication whether certain (parts of) software is being developed at another location. Implying a dependency on a proper network for real-time sharing and monitoring software or running applications.
	3	Amount of employees gives us insights towards the scale and possible amount of running applications inside an organization.
Software development	4	By knowing the amount of developers (and the relative % compared with total amount of employees), we can derive the role and importance a SD department has inside an organization.
	5	This denotes whether an organization runs a pipeline on premise and maintains it by themselves. At the other hand, if an organization does make use of such standardized deployment pipeline, we know that the Development Area is less important to them.
	6	The amount of deploying provides us with an indication how many times code is committed to main- and side branches and builds are being performed. This also gives an indication if an organization is applying CI, which is a suitable use case for applying containers.
Application & Infrastructure	7	The type of cloud that is used for infrastructure, application landscape, and solutions. Possibly on premise if an organization handles client-sensitive data.
	8	Stateless applications are one of the main business cases for containers. If an organization is already running stateless application, it is likely that either are already (partially) used, or the organization is suitable for applying containers. If not, based on the rough estimate, it should be found out what (amount of) stateful applications can switch to run stateless.
	9	If an organization is currently not using a microservices architecture style, the organization should have the ambition to migrate to such architecture style defined in their strategy plan for at least at the maximum of three years. Otherwise, using containers will not have a beneficial impact on business value.
	10	Applications that have older architectures may not be container (cloud) ready. If this percentage of potential applications is high, the organization should first rewrite their applications before making the switch towards containers.
	11	The last question provides us with an indication to what extent the organization possesses knowledge on virtualization. Having such knowledge available in the organization helps understanding the CMM. In addition, that knowledge will ultimately be of support when migrating to containers, as containerization aspects may be familiar.

Appendix X – CMM PDD tables

This appendix includes the corresponding activity and concept tables of the CMM's PDD (see section 10) according to Van de Weerd and Brinkkemper (2009).

Table 32: Activity table CMM PDD

Activity	Sub activity	Description
1. Execute Pre assessment	1.1 Assess the 11 questions of container-readiness	Perform the questionnaire of the Pre assessment. This includes the 11 questions that determine if an organization is container-ready. The results are used in PRE ASSESSMENT RESULTS.
	1.4 Evaluate results	Evaluate the results of the Pre assessment. The conclusion resulting from this step is used to determine whether an organization is container-ready, and is also part of PRE ASSESSMENT RESULTS.
2. Execute Maturity assessment	2.1 Assess Development Area	Perform the questionnaire of the MA's Development area. This includes questions that indicate the configuration of an organization's SDLC process and corresponding CICD functionalities.
	2.2 Assess Operations Area	Perform the questionnaire of the MA's Operations area. This includes questions that indicate the tasks an organization performs when their applications are released into production while running on containers.
	2.3 Assess Application & Infrastructure area	Perform the questionnaire of the MA's Application and Infrastructure area. This includes questions that indicate tasks and capabilities of an organization's application and infrastructure landscapes that provision the business operations.
3. Evaluate results	3.1 Evaluate assessment results	Evaluating all results of the Maturity assessment to discover how an organization is performing onto different areas inside their SDLC process, operations process, and CICD functionalities, in regards to containers.
	3.2 Determine topics of excellence	Determine on what topic(s) an organization excels regarding their virtual environment configuration inside the SDLC and operations process.
	3.3 Determine topics of improvement	Determine on what topic(s) an organization can improve regarding their virtual environment configuration inside the SDLC and operations process.
	3.4 Select to-be maturity level(s) of metric(s)	Determine the next steps of improvement of an organization to reach higher maturity levels on the specifically determined topics or areas.

Table 33: Concept table CMM PDD

Concept	Description
PRE ASSESSMENT RESULTS	Results of the Pre assessment that indicate whether an organization is container-ready.
DEVELOPMENT AREA RESULTS	Sub-assessment results of the Development area.
OPERATIONS AREA RESULTS	Sub-assessment results of the Operations area.
APP & INFRA AREA RESULTS	Sub-assessment results of the Application & Infrastructure area.
MATURITY SCORE	Assessment results of the Maturity assessment that includes all area results.
TOPIC OF EXCELLENCE	A topic in reference to SDLC an organization excels in. Can be one or multiple.
TOPIC OF IMPROVEMENT	A topic in reference to SDLC an organization can improve. Can be one or multiple.
RESULTS MATRIX	A document that contains the scores of the PA, MA, topics of improvements and excellence, and the desired maturity level improvements. The information of this concept is used in the BUSINESS CASE.
SELECTION OF DESIRED MATURITY LEVELS	The designed plan that guides an organization in improving their SDLC process and corresponding CICD configuration, , to ultimately achieve a higher maturity and (further) exploit containers.
BUSINESS CASE	The potential savings that can be realized if an organization switches from using VMs towards containers. The data is generated by the Business area.

Appendix XI – CMM linked to EAG

The topics of the CMM Maturity assessment are linked to EAG. In underlying table X, we described how the CMM content is linked to EAG. Stating such links shows how the artefact influences EAG from a technology perspective.

Table 34: Core agility attributes including technology-related sub-concepts (Sherehiy et al., 2007) linked to CMM

EAG attribute	Description
Flexibility	"*The ability to pursue different business strategies and tactics, to quickly change from one strategy/**task**/job to another.*" - Flexible organizational structures and practices (i.e. capacity of applications and infrastructure can be scaled or changed in balance depending on demands); - Workplace flexibility (i.e. all kinds of applications can be launched through virtual environments, independently of the location and infra agnostic).
	DEV *Component maturity – Functional*. This topic consists of metrics that provide specific functionality to the SDLC process which are related to containers. As these functionalities enable an organization to execute in other way, its flexibility in regards to SDLC is increased.
	OPS -
	AppInf *Landscape resilience*. Metrics regarding landscape resilience focus on immediate responses to landscape component failures and issues. Higher maturity refers to faster automatic responses which have larger impact to the application and infrastructure landscape.
Speed	"*The ability to complete requirements of all other agile characteristics in shortest possible time. The ability to learn, carry out tasks and operations and make changes in shortest possible time.*" - Performing tasks, operations, and making changes (i.e. fastness in launching instances of applications and infrastructure (application run time environments) to perform tasks and operations); - Time of: operations, production/service delivery (i.e. fastness of an application is able to start running a service).
	DEV *Task automation*. The extent how SDLC phase-specific tasks are performed automatically. The higher the maturity level, the more the phase-specific process is automated, and integrated into standard delivery pipeline. *Component maturity – Non-functional*. This topic refers to the components that are added to the SDLC process (*Functional*) that mature in improvements of the functionality (i.e. quality improvements). These maturity improvements are focused on the integration and containerization of the components.
	OPS *Lifecycle management*. The extent how the containers that support or contain applications are managed automatically increases speed of the SDLC process, as human interference is less required with higher automation. Higher maturity levels of this topic include more automated-related metric specifications
	AppInf -
Responsiveness	"*The ability to identify changes and opportunities and respond reactively or proactively to them.*" - Responsiveness to change in customer demands (i.e. scalability for application capacity); - Responsiveness to market and business environment changes and trends (i.e. rapidly integrating or executing new applications in organization's application and infrastructure landscape, in order to respond to or anticipate on market needs).
	DEV *Task automation*. When an organization notices a development or trend in its environment that requires

	rapid anticipation, a (highly) automated SDLC process is more able to provide such anticipation (e.g. develop new or modify current application, complement application with a new service etc.).
	OPS -
	AppInf *Scaling potential*. Depending on customer or user demand, an organization can scale-up/-down their infrastructure and applications. Higher maturity leads to more component usage that are enabling rapid and automatic scaling of required services or hardware instances. This supports improved responsiveness. *Network maturity*. A higher maturity in an organization's network of infrastructure, the more stable the network is to handle and manage increases in performance, scale up and down,
Integration and Low complexity	*"Close and simple relations between the individual system components, easy and effortless flow of the materials, information and communication between the system components, organizational structures, people, and technology."* - Integration of people, technology, and organization (i.e. integration of DevOps runtime environments of applications by the means of containers, while using the IaC principle); - Synthesis of diverse technologies, skills, and competencies (i.e. combination of microservices architecture with containers and IaC); - Low complexity of structure, relationships between structure elements (i.e. technical architecture, applications/services that are independently managed with determined runtime environment); - Flow of material, communication and information between different organizational structures and systems components (i.e. networking maturity, process maturity, API configuration); - Enhanced interaction between processes.
	DEV -
	OPS *Adminstrative tasks*. This topic consists of the Logging metric.
	AppInf *Network maturity*. Network techniques enhance fast and responsive internal networking and communication. A lot of solutions exist that provide such techniques. A new technology that foster low complexity in internal networking is the API technology. Hence, we incorporated this technology in the maturity levels, and linked this topic to this EAG attribute.
Culture of change	*"A description of environment supportive of experimentation, learning, and innovation and is focused on the continuous monitoring environment to identify changes. Culture of change is an environment where people on all organizational levels have positive and fearless attitude to changes, different opinions, new ideas, and technology."* We did not incorporate this core attribute into the relationship to both containers and VMs, as the CMM. The reason is that this attribute is focused on people (O of OPT), and is therefore according to us not directly related to our aim of technology (T of OPT).

Appendix XII – TD: Expert interview results of Iteration 2

Appendix XII, XIII, and XIV elaborate on the results of the performed expert interviews of respectively iterations 2, 3, and 4. Each interview report consists of a short introduction about the interviewee, possible quotes ("*quote 1*"), a table including summarized, relevant feedback towards CMM modifications and our rationale, a legend about the tag visual[15], and a figure of the tagged soundbar. In the tables, some sentences are placed between square brackets. This indicates a statement of questions from us, where the interviewee answers to.

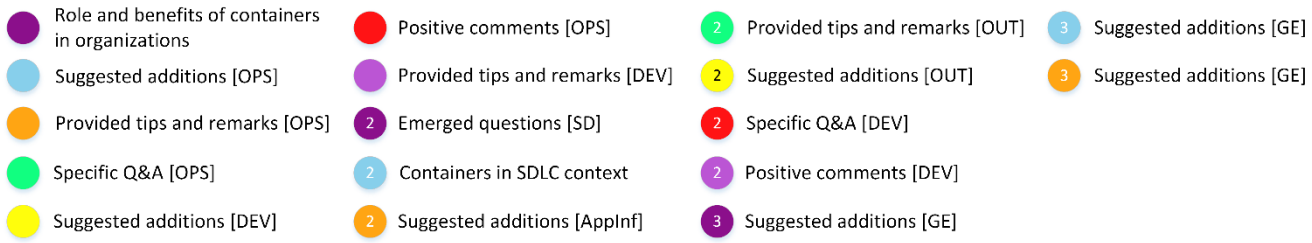
No, rejected	Uncertain, perhaps future	Yes, but for future	Partially implemented	Implemented

The interviewees (experts) have been partly anonymized due to made confidentiality agreements. Their input and quotes are removed from the public version of the thesis.

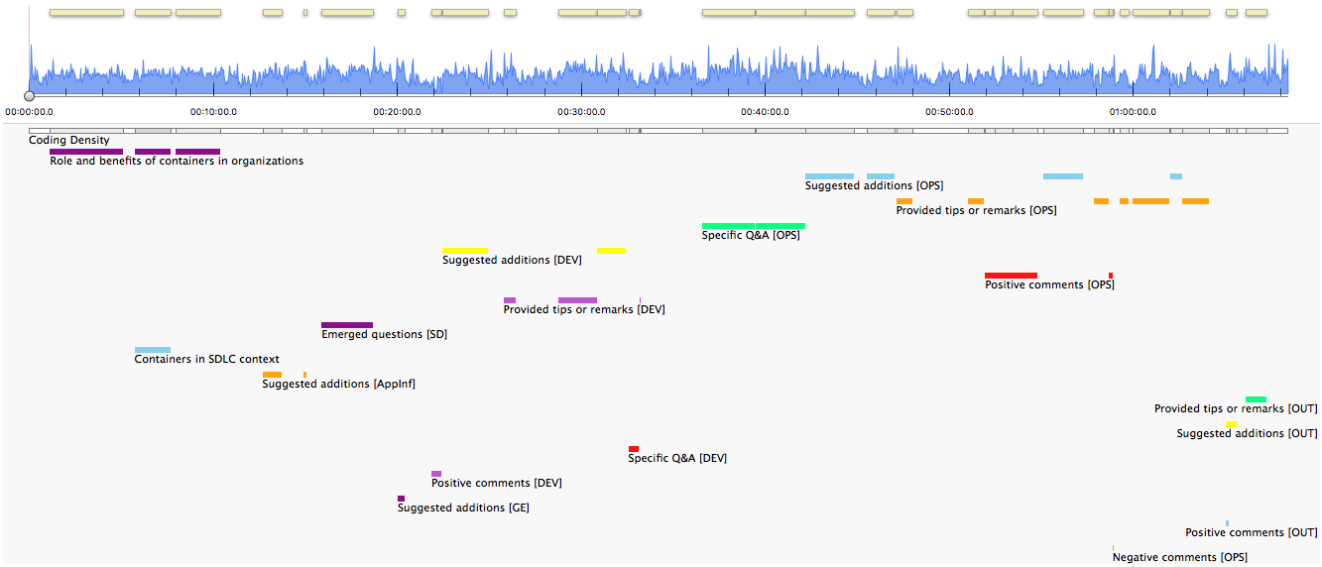
Results of expert interview – IT2Int#1

Table 35: Results

#	Summarized description of main findings	Rationale	Y/N
1		We agree and incorporated a question into the PA about the specific cloud type an organization is using.	
2		As this metric is focused on container usage in all tasks related to coding, we agree with this statement of the expert and incorporated it into the CMM.	
3		Monitoring of the applications that have been developed is also an aspect of SDLC, therefore we agree that we should incorporate this metric. We partially implemented this metric, since it is a phase, and we are not sure in which area to implement.	
4		We believe there are differences in how containers are managed and used when infrastructure and/or a pipeline is managed in-house or taken from the cloud. However, this is too comprehensive to process in current project. Therefore, this is an aspect for future work.	
5		We agree and processed this into the CMM with its current structure.	
6	[statement]		-



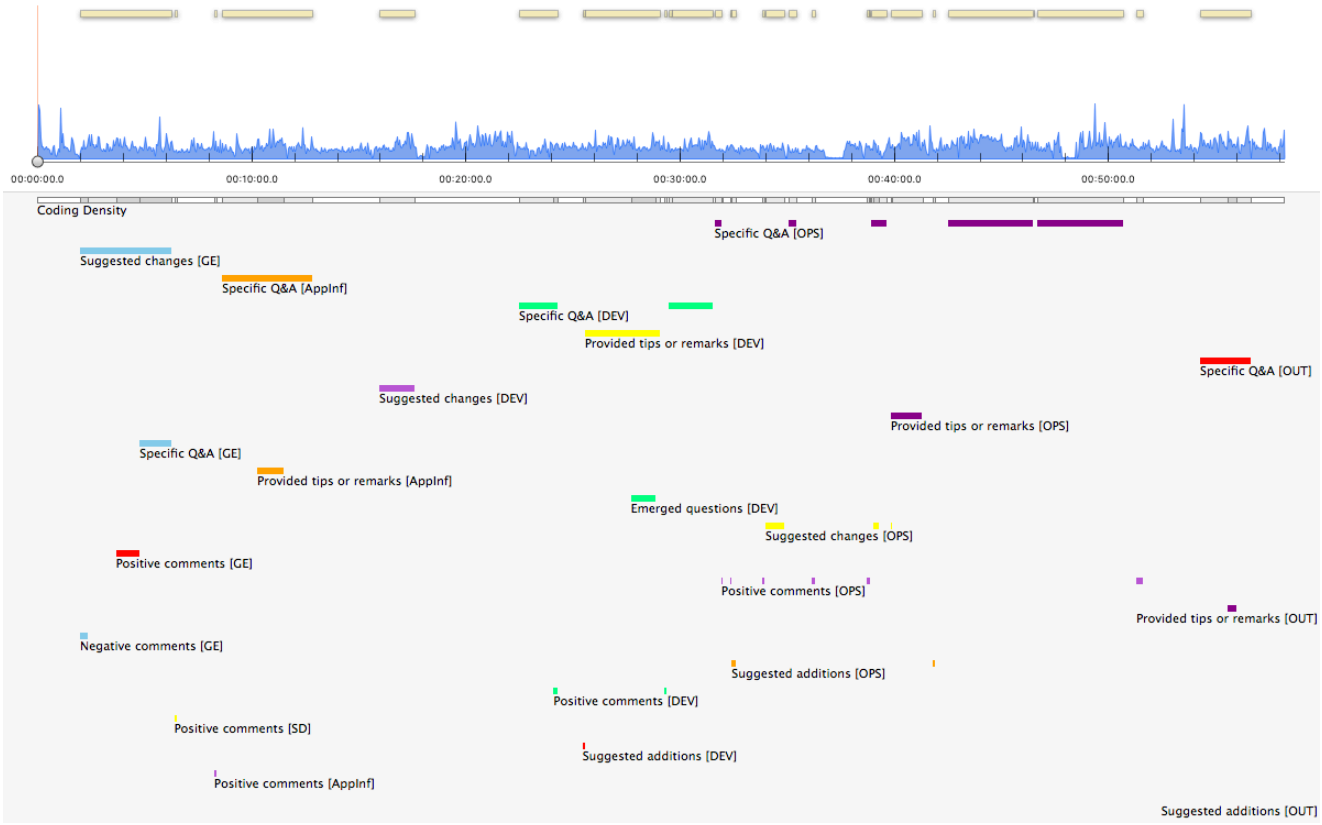
[15] Each interview had more than ten discussed *aspects* of the taxonomy. NVivo 12, however, is restricted to depict seven colors. Therefore, we numbered the tag legends in cumulative manner after each cycle of seven colors, to increase overview.



Results of expert interview – IT2Int#2

Table 36: Results

#	Summarized description of main findings	Rationale	Y/N
1		We agree and changed the question to SD processes only.	
2		Valid point, perhaps for future research.	
3		Private cloud usage gives an indication that an organization may be of risk for performance loses or lack of storage. However, this depends on the context and operations of an organization. Therefore, we want to include this question in the PA.	
4		We agree with this statement as every tasks generates value. The difference is that no all tasks directly generated business value. Therefore, we incorporated the sharpening of the concept in into the model	
5		We argue that this is a potential outcome for the highest levels of the metrics	
6		We added a Monitoring phase in the Development area of the CMM. By doing this, it follows the SDLC phase structure.	
7		This may be a useful addition for when developing a further refined version of the MA questionnaire. As in the current research, we focus on developing a first version of the CMM.	
8	[statement]		–
9		Specified tests.	
10		Changed hierarchy.	
11			



Results of expert interview – IT2Int#3

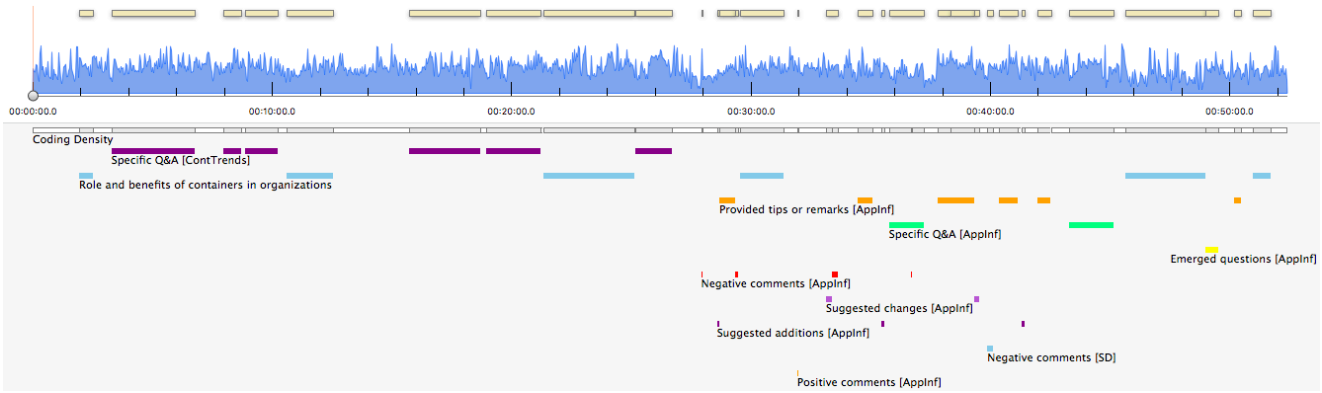
Table 37: Results

#	Summarized description of main findings	Rationale	Y/N
1		Processed into metric specifications.	
2		Based on the (functional) characteristics of containers, we agree with this statement and processed this into the PA.	
3		Based on the response, we agree with this statement.	
4		We mostly agree with this statement and shortened the PA. However, we think that incorporating other questions regarding an organization's applications, (MS-) architecture, and frequency of deployment is also valuable.	
5		We defined a question in the PA about such deployment pipeline usage. However, we do not agree with the statement about CMM irrelevance. Organizations still have to manage their internal operations to be able to use those services. Therefore, we believe the CMM is still relevant in such cases.	
6		We created a grading scheme for the PA. For future research, at the moment when the cross-links are defined, we think it is interesting to create a grading scheme for the MA as well.	
7		Monitoring is removed from DEV area and processed into the OPS area. We did not remove it from the model, since we think that monitoring the applications is important for the SDLC process, thus for the CMM.	
8		We disagree with this statement. We want to maintain the level of detail in the differentiating levels of skill regarding container usage in SDLC, we are aiming to provide to users of this maturity model.	
9		Findings such cross-links would be a major improvement to the CMM, especially in the results part. Although, this development is not suitable for current research. Therefore, include this feedback into future works.	

Results of expert interview – IT2Int#4

Table 38: The feedback from findings

#	Summarized description of feedback	Rationale	Y/N
1		We agree. We tried to elicit more answer by providing too broad questions. However, we noticed that this did not improve answers.	
2		We added aspects regarding MSA to the PE. We agree and believe that an organization that wants to benefit from containers, should consider to migrate to a MSA.	
3		As this architectural change impacts both applications and how organizations work with them, or develop them, we agree to process this aspect into the PA.	
4		It would be an informative addition to the model, but we determined not to incorporate this, as the PA's decision should be based on an organization's existing functionalities and opportunities in the SDLC configuration, not on its (business) motives.	
5		Interesting to know the opinion from experts directly from the field of practice. However, not for deciding about container-readiness.	
6		Changed grading of the PA.	
7		Not relevant in current form of questioning.	
8		Conducted more research on this topic and processed results in the CMM.	
9		Agreed and processed.	
10		We think this is a too detailed aspect for current research project. We do believe this concept can play a role in future research, as performance gains can be won by using the correct (container-minded) languages	
11		Processed all CICD related metrics to specific phases of SDLC.	
12		Done.	



Appendix XIII – TD: Expert interview results of Iteration 3

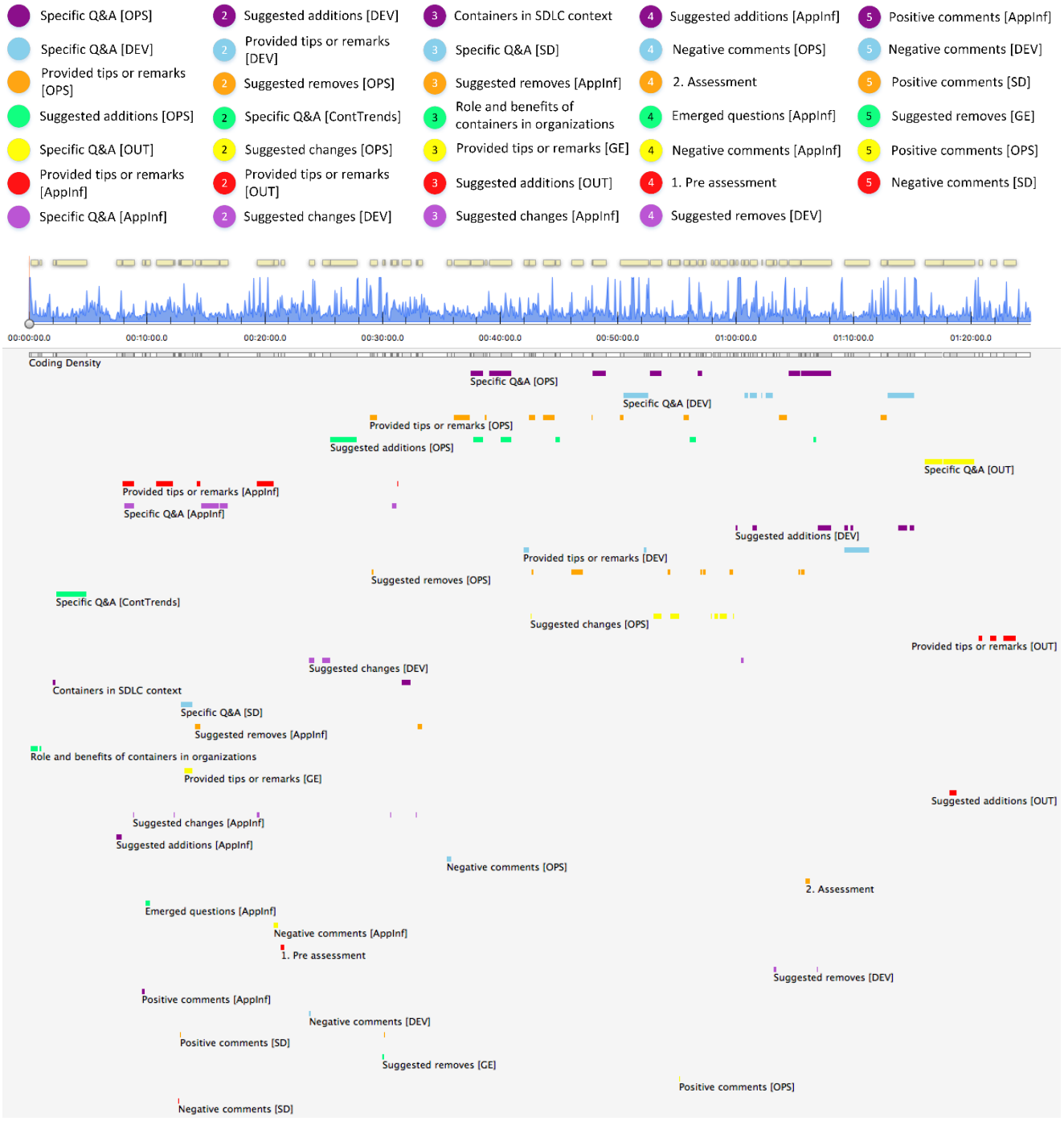
The interviewee is partly anonymized due to made confidentiality agreements. Its input and quotes are removed from the public version of the thesis.

Results of expert interview – IT3Int#5

Table 39: Results

#	Summarized description of main findings	Rationale	Y/N
1		Agreed with both statements. Shortened the PA and added a question about the amount of stateless applications (%) of all applications.	
2	[statement]		–
3	[statement]		–
4		We agree and changed the question in the PA about a pipeline towards a 'standardized deployment pipeline for more than 50% of your SD processes'.	
5		Outsourcing may be interesting to know, since it can indicate whether an organization needs a strong network, the amount of dependencies on external dev-teams and thus how CI is configured. We decided not to include this in the current CMM, as it would take too much time to investigate this link. Therefore, we added this to the future works.	
6		Verified our programming languages question from the prior expert interview.	
7		We agreed with this statement. The business motive is interesting to know, but will not be the major aspect that influences the decision whether an organization is container-ready.	
8		We understood the statement and processed everything. Moreover, this is the same feedback from IT2Int#4, which we misunderstood and did not perceive as relevant for the CMM at that moment.	
9		We understood the statement and switched monitoring from DEV area to OPS area. We also added the proposed concepts as metrics to the CMM in OPS. The images are added as metrics in the DEV area.	
10		Agreed. We processed all test-related metrics to the test phase in DEV area. We removed the split of app-dependent and –independent tests, and specified all tests through automation.	
11		These levels of maturity in automation described how automatic tasks are executed. We used this level structure as a base for defining each phase's Task automation metric, where we customized the m-levels.	
12		We also used this (abstract) level structure as a base for all our component metrics. For each metric, we refined the levels suitable to its functionality.	
13	[statement]	This validates our statement by a relevant expert that is specialized in the domain of interest.	–
14		Agreed and processed into CMM.	
15		We think that this would be an addition of high value, as we can further specify each metric into more detail, and can withhold information for future dashboards. However, considering our scope and remaining time, we considered this would require too much time. Hence, we believe this is an aspect for future research.	
16	[statement]		–

As this expert is specialized in the exact domains the CMM is aiming for, we regard this interview as the most valuable interview of all iterations. The interviewee had suggestions, criticism, and positive feedback on both high and specific level components. Therefore, where possible we processed all of his feedback into the model.


Specific Q&A [OPS]
Specific Q&A [DEV]
Provided tips or remarks [OPS]
Suggested additions [OPS]
Specific Q&A [OUT]
Provided tips or remarks [AppInf]
Specific Q&A [AppInf]
Suggested additions [DEV]
Provided tips or remarks [DEV]
Suggested removes [OPS]
Specific Q&A [ContTrends]
Suggested changes [OPS]
Provided tips or remarks [OUT]
Suggested changes [DEV]
Containers in SDLC context
Specific Q&A [SD]
Suggested removes [AppInf]
Role and benefits of containers in organizations
Provided tips or remarks [GE]
Suggested additions [OUT]
Suggested changes [AppInf]
Suggested additions [AppInf]
Negative comments [OPS]
2. Assessment
Emerged questions [AppInf]
Negative comments [AppInf]
1. Pre assessment
Suggested removes [DEV]
Positive comments [AppInf]
Positive comments [AppInf]
Negative comments [DEV]
Positive comments [SD]
Negative comments [DEV]
Positive comments [SD]
Suggested removes [GE]
Positive comments [OPS]
Negative comments [SD]
00:00:00.0
00:10:00.0
00:20:00.0
00:30:00.0
00:40:00.0
00:50:00.0
01:00:00.0
01:10:00.0
01:20:00.0
Coding Density

Appendix XIV – TD: Expert interview results of Iteration 4

This appendix provides the main feedback derived from all interviewed experts of IT4. We did not use Nvivo12 to analyse the interviews, but made notes during the interviews instead, which we afterwards processed into the tables.

The interviewees (experts) have been partly anonymized due to made confidentiality agreements. Their input and quotes are removed from the public version of the thesis.

Results of expert interview – IT4Int#6

Table 40: Results

#	Summarized description of main findings	Rationale	Y/N
1		Added a question about applications that run virtualized.	
2		Processed the feedback into grading of PA.	
3		We agree and processed this information into our results.	
4		We based the first version of our CMM on literature. Because of this, we missed the delivery aspect. After further investigation on the concept, we understood this statement and immediately started to process this feedback into the CMM. We added an area regarding the application and infrastructure landscape of an organization to provision its business domain.	

Results of expert interview – IT4Int#7

Table 41: Results

#	Summarized description of main findings	Rationale	Y/N
1		Partially implemented as we complemented the metrics specification.	
2		Agreed and added the metric and the maturity levels.	
3		Processed these specifications towards the metrics.	
4		Partially processed feedback into corresponding metrics.	
5		For now, we did not incorporate this, as we maintained the metric. However, we consider this as valid feedback and therefore may include it in future versions.	

Results of expert interview – IT4Int#8

Table 42: Results

#	Summarized description of main findings	Rationale	Y/N
1		We agreed with this suggestion, as this helps users in understanding the cohesion of the model. Due to this change, users will associate the model quicker towards something they know. Hence, improving the usability from their perspective.	
2		This feedback is processed into the App&Infra area metric around service orchestration. The maturity specifications of level 3 and 4 are complemented with this feedback. Therefore, partially implemented.	
3		Added Vulnerability scanning to the metric specification of level 3.	
		We added A/B testing to the Load balancing metric. Additionally, we added On-premise as infrastructure option in level 5.	

4		This would be a great form to convert the final version of the CMM into order to show the content of the maturity levels in a brief way to (higher) management. However, due to time restrictions and current content of the CMM, we did not include such abbreviated version in this thesis. Future research should focus on this aspect.	

Results of expert interview – IT4Int#9

Table 43: Results

#	Summarized description of main findings	Rationale	Y/N
1		We added this concept in the CMM, as we argued that this functionality could be a level 4 or 5 maturity specification, as the impact of such function on a system tests the resilience on high level.	
2		A valid point that improves quality of both the model as the research process.	
3		We agree with this addition, as microservices is the main architectural style that enables the benefits of containers in an organization. Extending the migration to a MSA towards a long term strategy (more than three years), means that stateless applications are not important to the organization. Hence, containerization does not improve that organization, making the CMM irrelevant for them.	
4		We immediately processed these suggestions into the CMM, as we defined these metric specifications in collaboration with this expert.	
5		The same as in row 4, we determined these high-level metric specifications together with the expert during this interview. Partially implemented.	
		Equivalent to row 4 and 5, we stated and directly incorporated these metric specifications into the metrics. Later on in this iteration, we made some adjustments to these metric specifications. Hence, we partially implemented this feedback.	
6		We immediately processed these suggestions into the CMM, as we defined this metric specification in collaboration with this expert.	

Results of expert interview – IT4Int#10

Table 44: Results

#	Summarized description of main findings	Rationale	Y/N
1		This is true. We eventually processed this container usage as component-functionality throughout the metrics in the metric specifications (maturity levels), and removed the specific container usage per phase.	
2		We believe this is interesting to know for dev-teams that developed an application, and for management of the development department as new strategies can be applied. However, we intend to include technology-based aspects only into the model (besides metrics that include human-interaction as metric specification). This feedback is denotes responsibility of actual dev-teams.	
3		In collaboration with the expert, we defined five levels for the Build server metric.	
3		Agreed with this statement. At first, we added for each DEV area phase a metric that consists of business logic container usage. After that, we determined to process this as a high-level metric in the App & Infra area.	

Results of expert interview – IT4Int#11

Table 45: Results

#	Summarized description of main findings	Rationale	Y/N
1		In several level 5 metric specifications, we included the ecosystem aspect as this realizes further reach for functionality in the environment of an organization.	
		We added to level 3 of Artefact repository – system specifications 'central', and the reusing functionality.	
2		We agreed and complemented the metric with these additions.	
		We split both unit-testing and Integration testing. Of Unit-testing, we have put the internal sources at level 2, as internal sources are easier to reach and control mostly. The unit-testing of external sources is added to level 3. For Integration, we use the same reasoning and processed internal into level 3 and external into level 4.	
3		Valuable suggestion. Decided to postpone this feedback to future versions.	

Appendix XV – TD: Interview protocol IT2

Semi-structured interview with internal PwC expert

Goal: to find out what experts think about the content of a Container Maturity Model.

Introduction

- Introducing myself;
- Tell about the master;
- Briefly explain the master thesis project, including structure, deliverables, expectations;
- Explain the goal, structure, subjects, and timespan of this interview, and why this person;
- Ask for permission to record this interview. The record will only be used to make a transcription and analysis about the results. The analysis is used to scientifically use the results to further design the CMM.

Start recording with phone

General

"**I would like to start with some general information.**"

- Ask interviewee to tell something about him-/herself, position, daily tasks etc.;

- What do you think about the role of containers in organizations nowadays?
- What do you think is one of the main benefits for organizations of using containers?

explain about found results on containers showing their benefits solely in certain contexts

- To what extent do you agree with this? What do you think about the SDLC context?

Container Maturity Model (CMM)

Explain about the structure of the model, specifying the three areas (Containers, How Agile, and Business Case).

In general, ask the expert about his opinion about:

- Relevance of element;
- Elaboration of element in current model;
- Agreements (positive comments);
- Missing elements regarding SDLC AND containers (negative comments).

1. Pre assessment (10 min)

- Considering the goal of CMM, to what extent do you think that the content of the **pre assessment** is sufficient to determine if an organization is container-fit (i.e. will benefit from applying containers)?

- Do you see any topics that are crucial for determining the container-fitness?

2. Assessment (45 min)

Walk through the model by discussing every step of all three areas (Development, Operations, Output).

Development (20 min)

- What do you think about the metrics that are currently measured?
- To what extent do you agree with the given levels, and their positioning?
- Would you add or remove elements from this CMM area?

Operations (20 min)

- What do you think about the topics that are currently questioned to measure the organization's agility of their SDLC configuration?
- What do you think about the given levels of these topics?
    - Are they using the correct concepts?
    - What do you think about the order?
    - To what extent do you agree with the given maturity levels (e.g. what would you consider mature)?
- Given the goal of this CMM area, would you add or remove elements to increase the effect of this area?

Output (5 min)

- What do you think about the content of this CMM area?
    - Do you agree with the statement about "*VMs vs Containers*"?
    - And that the topic of main benefit of containers is storage and data usage?

Finalizing (5 min)

- Is there something I forgot to ask about, what you consider as important?
- Do you want to add something else?

***End of interview**

- Thank interviewee for the interview;
- Offer to send the results when project is finished;
- Ask for possibility of additional questions if later required;
- Ask if interviewee wants to see the transcript to check validity.

Appendix XVI – TD: Interview protocol IT3 & IT4

Semi-structured interview at ING

Goal: to find out what experts think about the content of a Container Maturity Model.

Introduction

- Introducing myself;
- Tell about the master;
- Briefly explain the master thesis project, including structure, deliverables, expectations;
- Explain the goal, structure, subjects, and timespan of this interview, and why this person;
- Ask for permission to record this interview. The record will only be used to make a transcription and analysis about the results. The analysis is used to scientifically use the results to further design the CMM.

Start recording with phone

General (10 min)

"**I would like to start with some general information.**"

- Ask interviewee to tell something about him-/herself, position, daily tasks etc.;

- What do you think about the role of containers in organizations nowadays?
- What do you think is one of the main benefits for organizations of using containers?

explain about found results on containers showing their benefits solely in certain contexts

- To what extent do you agree with this? What do you think about the SDLC context?

Container Maturity Model (CMM)

Explain about the structure of the model, specifying the three areas (Containers, How Agile, and Business Case).

In general, ask the expert about his opinion about:

- Relevance of element;
- Elaboration of element in current model;
- Agreements (positive comments);
- Additional tips or remarks;
- Missing elements regarding SDLC AND containers (negative comments).

1. Pre assessment (20 min in IT3 – 10 min in IT4)

- Considering the goal of CMM, to what extent do you think that the content of the **pre assessment** is sufficient to determine if an organization is container-fit (i.e. will benefit from applying containers)?

- Do you see any topics that are crucial for determining the container-fitness?

2. Assessment (80 min in IT3 – 45 min in IT4)

Walk through the model by discussing every step of all three areas (Development, Operations, Output).

Development (30 min in IT3 – 20 min in IT4)

- What do you think about the metrics that are currently measured?
- To what extent do you agree with the given levels, and their positioning?
- Would you add or remove elements from this CMM area?

Operations (40 min in IT3 – 20 min in IT4)

- What do you think about the topics that are currently questioned to measure the organization's agility of their SDLC configuration?
- What do you think about the given levels of these topics?
    - Are they using the correct concepts?
    - What do you think about the order?
    - To what extent do you agree with the given maturity levels (e.g. what would you consider mature)?
- Given the goal of this CMM area, would you add or remove elements to increase the effect of this area?

Output (10 min in IT3 – 5 min in IT4)

- What do you think about the content of this CMM area?
    - Do you agree with the statement about "*VMs vs Containers*"?
    - And that the topic of main benefit of containers is storage and data usage?

Finalizing (10 min in IT3 – 5 min in IT4)

- Is there something I forgot to ask about, what you consider as important?
- Do you want to add something else?

*End of interview

- Thank interviewee for the interview;
- Offer to send the results when project is finished;
- Ask for possibility of additional questions if required later;
- Ask if interviewee wants to see the transcript to check validity.

Appendix XVII – Container Maturity Model

This part is excluded from the public version of the master thesis. Please feel free to contact me for additional details about specific topics or concepts.

Appendix XVIII – TD: Sources and traceability of metrics

Included in this document is an Excel sheet that provides an overview about the origin of each metric, and modifications of all metrics.

Sources and traceability of

Virtualization technologies for Enterprise Agility

Developing a Maturity Model for the application of containers in organizations

Tom Osinga

Master Business Informatics, Department of Information and Computing Sciences, Utrecht University, Utrecht, the Netherlands

t.h.m.osinga@students.uu.nl

Abstract. Organizations are nowadays working on agile transformations in order to be able to anticipate on all emerging trends in their environment. As IT is becoming more important for throughout all business and supporting operations, organization can weaponize themselves by focusing on making their IT more adaptive or agile. Containers are an emerging technology that have the potential to support organizations in becoming more agile in respect to business and IT. However, from a literature perspective, containers are not denoted as being able to support in becoming agile, nor posses most traditional organizations knowledge on applying containers. This study aims to fill this scientific gap of knowledge, and fulfil the call from industry by designing a Container Maturity Model (CMM). We selected the use case of SDLC to focus on. The CMM was developed through executing multiple iterations where we interviewed experts with varying specializations. As a result, we delivered a maturity model that consists of three parts, including a Pre assessment (PA), Maturity assessment (MA), and a Results matrix. Organizations can use the CMM to start applying containers, or improve their container utilization based on the provided differentiating levels of skill regarding containers and SDLC. Nevertheless, we believe future research is required to further develop, validate, and test the maturity model.

Keywords: Enterprise agility; Virtualization Technologies; Containerization; Maturity Model; Software Development Lifecycle; Microservices Architecture.

1 Introduction

Organizations nowadays have to become increasingly more agile to anticipate on all emerging trends in their environment. Change is motivated from different needs that are difficult to predict. Factors such as hyper-competition, increasing demands from customers, regulatory changes, and technological advancements make it an important

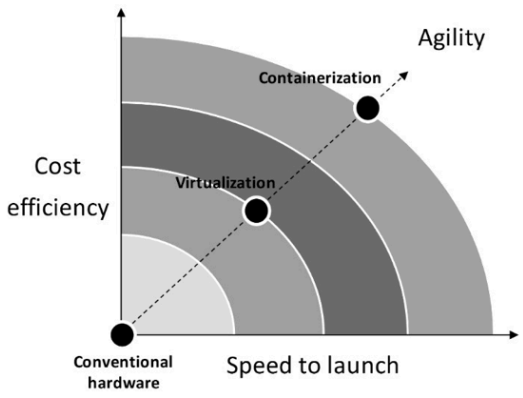
adfa, p. 1, 2011.

determinant of firm success [1]. Kaddoumi and Watfa [2] argue that businesses facing changing environments require their business and IT executives to act fast and quick towards uncertainty and unpredictability, in order to be able to accept and adapt to such changes. To successfully deal with unpredictable, dynamic, and constantly changing environments, many different solutions have been proposed, such as networking, reengineering, modular organizations, virtual corporations, flexible manufacturing [3]. Through the years, these developments led to the incremental growth of enterprise agility (EAG), which is agility focused on all departments of an organization.

Currently, agility concepts are not yet explicitly defined and conceptualized, and there is no commonly accepted definition of enterprise agility in the scientific domain [3]. Nonetheless, core agility attributes have been identified, which are flexibility, responsiveness, speed, culture of change, and integration and low complexity. These core attributes can be translated to the main enterprise components, which are Organization, People, and Technology (OPT) [3]. We found research towards agility regarding the O and P domains, however, we did not find specific research on agility regarding technology. This denotes a gap of knowledge from the scientific perspective regarding Technology and EAG.

However, a technology that explicitly enabled agility in infrastructure is virtualization. Virtualization made it possible for organizations to virtually partition their infrastructure, enabling them to use the same infrastructure components for multiple goals instead of a single goal. The first form of virtualization was introduced in the late 1960s. Nowadays, virtualization technologies are becoming more widely used, as large data centers exist where multiple virtualization technologies are making abstractions of physical hardware, and create aggregated pools of computing resources [4].

Around 2013, Docker popularized the concept of *containers*, a further advanced virtualization technology. In short, Docker containers consist of an executable package for pieces of software, which means that all required files and components are available in the container to be able to run on the users' (Host) OS, and easily create a virtual environment to provision business operations. Compared to VMs, containers are rationalized (automated) hardware deployments consisting of a lightweight OS. Due to their 'leanness', containers are more aligned with the aforementioned agility core attributes than conventional virtualization solutions. Figure 2 visualizes this statement about containers and their potential to support agility.



Containerization is a success in the Cloud-Enabling Technologies (CET) market. Organizations who spend $500,000 or more a year on container technologies has increased to 32% up from five percent in 2016 [5]. Another study states that the overall container revenue was $495 million in 2015, and already increased with 123,64% to $1,107 million in 2017, and will be approximately $2,2688 million in 2020 [6]. This denotes the potential and success of the technology.

Although, through expert interviews, despite that we found that the field of practice is aware of containers' benefits, we found no sources that scientifically define containerization as a technology that can be used to pursue enterprise agility. This finding also denotes a gap of knowledge regarding Technology and EAG. Moreover, we additionally found two reasons as need for knowledge from the field of practice: 1) (traditional) organizations desire tangible knowledge on how to start applying containers, or 2) knowledge on further improving (i.e. maturing) their current container utilization. Mainly in reference to the use case of SDLC. Besides the gap of knowledge, this emphasized the need for knowledge from the field of practice.

The role of IT is increased to a primary function, as IT is integrated in other functions of organizations. This implies the importance of efficient management of IT. Enterprise architecture (EA) provides insights about an organization's overall cohesion and dependencies of all (architectural) elements, including managing organizational aspects, application landscape, and infrastructure. This aligns with the aforementioned OPT components. Virtualization technologies enables organizations to increase the utilization of their infrastructure, whereas containers bring equivalent advantages, although faster, require fewer resources, and through different functionality. Therefore, the benefits of containers have a stronger alignment with the core agility attributes, and can lead to a relation with T of OPT. There is, however, no explicit link established between containers and EAG. This corresponds with the aforementioned gaps of knowledge.

In this study, we aim to fulfil the discovered gaps of knowledge. When we established such a link between EAG and containers (T), organizations that aim to become enterprise agile can use this link to discover to what extent containers are applicable for their goal. Subsequently, EA should be applied to support the integration of containerization in the organization, by depicting its overall cohesion and dependencies. After that, containers can provide their the agile-enabling characteristics. Therefore, we designed a container maturity model (CMM) with focus on SDLC, that consists of differentiating levels of skill regarding containers. This model provides basic knowledge on starting to use containers, as well as further improving container utilization. Finally, it states a relationship between Technology and EAG.

2 Related works

Some literature exists that related technology to agility. One study defined the importance of technology for enterprise agility. This study related attributes and practices

of an agile organization regarding technology, which are 1) technology awareness, 2) leadership in use of current technology, and 3) skill and knowledge enhancing technologies [7]. Another study developed a conceptual model of enterprise agility that encompasses of agility drivers, capabilities, and providers. When referring to technology, they state that *Technological innovations* is one of the main agility drivers, and *information integration (infrastructure)* is one of the important agility providers [8]. Also, Pal and Pantaleo [9] state that organizations should configure adaptive infrastructures.

Concluding, these studies describe the importance of being aware of new technological trends and being able to skillfully exploit these new trends by having an adaptive infrastructure. Considering the rapidness of current developments in technology, the role of technology in enterprise agility is bound to increase.

Moreover, due to an increasing demand for agile software solutions, more studies were conducted towards adaptable IT landscapes. As a result, the following characteristics were introduced and linked to agile technology:

- Technical integration [9, 10, 11, 12, 13];
- Standardization [3, 9, 11];
- Reusability [3, 8, 14];
- Scalability [3, 8, 14].

These characteristics align with a modern architectural pattern: the microservices architecture (MSA). MSA is an approach for developing and managing a single application as a *suite of small services*, each running its own process and communicating with lightweight mechanisms [15]. This MSA pattern realizes above characteristics of adaptable IT landscapes. In addition, as the MSA decouples monolith architectures and is also focused on working with short-lived, stateless instances of applications and services, it meets the specifications containers require before their advantages can be exploited. Hence, the MSA pattern realizes the use case-specific container demands (portability, stateless, resource efficient) in an EA. Therefore, integrating containers in an organization's EA while applying the MSA pattern increases EAG in regards to technology.

3 Method

The Design Science research method of dr. R. Wieringa is selected as main research method. Design science is the design and investigation of artefacts in context [16]. We chose this method as it combines descriptive research with exploratory research, in order to design an artefact to improve the problem context. Our aim was to discover the opportunities of combining agile concepts with virtualization. Subsequently, to use EA to position containers in an organization, to ultimately support enterprise agility from a technology perspective.

In the Problem Investigation phase, we conducted a literature review and expert interviews. We aimed to answer the first RQs of our study. These RQs were focused on

stating the state of the art of enterprise, virtualization, and containerization. Consequently, we validated our findings with the experts, and additionally questioned their perspective on container utilization. In addition, we explored how EAG and virtualization technologies are linked to each other.

Based on the results, we proposed to design the CMM, focused on SDLC, and to further conduct research on SDLC and maturity models. Ultimately, to enhance EAG from a technology perspective.

3.1 Treatment Design

We defined the following hypotheses:

> **H_0**: *Integrating containers in the SDLC process does not increase support for organizations' enterprise agility regarding business and IT.*
>
> **H_1**: *Integrating containers in the SDLC process does increase support for organizations' enterprise agility regarding business and IT.*

To test these hypotheses, we executed different types of expert interviews through four iterations. We intertwined the design process with validation sessions, so that we were able to gather additional knowledge on SDLC and perceive feedback of the experts regarding our CMM. Hence, each iteration used a different CMM concept version to present the experts, resulting in a further developed CMM version.

We chose the form of multiple iterations for designing the CMM, in order to harvest varying types of information. Each iteration had a different setting and goal. IT1 consisted of brainstorm sessions and derived information from the prior PI results. The remaining iterations consisted of semi-structured expert interviews with internal PwC experts, and external experts from a large international bank. This collection of varying information acted as our main source to design the CMM from. Hence, all iterations combined depict a design process of incremental nature.

- *Iteration 1*: Designing the CMM based on all research conducted during Problem Investigation phase.
- *Iteration 2*: Designing and validating the CMM based on feedback from internal PwC experts.
- *Iteration 3*: Designing, validating, and testing the CMM based on feedback from external experts.
- *Iteration 4*: Additional designing and validating the CMM based on feedback from both internal and external experts.

3.2 Data analysis

The objective of this design process is to develop a maturity model regarding the utilization of containers in the SDLC process, which is validated by experts from the

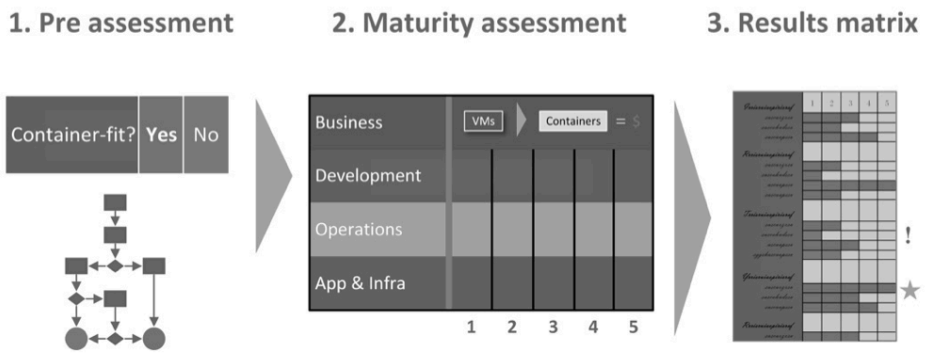
field of practice. We recorded the expert interviews of iteration 2 and iteration 3. Subsequently, we analyzed the interviews using NVivo 12. This is a tool to analyze transcriptions and audio files of recorded interviews.

We defined a taxonomy to denote the subjects we focus on in the analysis. Topics denote the context of discussion that we aimed for to focus on. Our focus is positive and negative comments of experts, next to their suggested changes. These are called as *Aspects*. These kind of aspects support the verification by experts of the CMM. Aspects refer to statements about metrics from the experts. Using the taxnomoy, we 'tagged' periods of time in the audio files (sound bars). The taxonomy is shown in Appendix D.

4 Results

The CMM assesses an organization's container-readiness, and the maturity of their SDLC process. The latter also focuses on the application landscape and infrastructure, and corresponding business value. Different areas consist in the CMM, whereas each area includes several metrics. These metrics are components that are used in the SDLC process with respect to containers, and are defined throughout five different levels of maturity. The CMM comprises three main parts:

1. *Pre assessment* (PA) – determines whether an organization is container-ready;
2. *Maturity assessment* (MA) – the Maturity assessment that assess an organization's SDLC configuration with the focus on containers;
3. *Results matrix* – depicts the results of the MA, including the maturity levels and business output.



4.1 Pre assessment

The Pre assessment exists of 12 questions that test an organization's container-readiness. Based on the decision, it is denoted whether an organization is benefitted by containers, and hence should continue with the MA. The 12 question are grouped into three

areas, which are General, Software development, and Application and Infrastructure. We provided the questions in Appendix A.

4.2 Maturity assessment

The Maturity assessment exists of four Areas, each having its own focus. Provided below are these areas, accompanied with an explanation:

1) *Business* (Output) [OUT]: results and potential savings by switching to containers (i.e. business case).
2) *Development* [DEV]: contains main phases of SDLC, including phase specific CICD components.
3) *Operations* [OPS]: exists of all tasks that are performed when containers are released in production.
4) *Application landscape & infrastructure* [AppInfr]: the hard- and software components that facilitate support for the SDLC process.

We chose to use the maturity level *names* of Dreyfus and Dreyfus [17], as we think these names are better suited to use in our CMM. Dreyfus denotes names for the learners of a skill and describes behavior per maturity level, instead of implying functionality requisites as CMMI [18] does. Appendix B provides all tables that include the metrics of each area.

4.3 Results matrix

The result matrix shows the results of the maturity of an organization. Per area, topic, and aspect, the maturity score is given from which one can derive and select its topic(s) of excellence and topic(s) of improvement. With this information, organizations can orientate themselves on improving their overall SDLC maturity in regards to containers.

4.4 Process Deliverable Diagram of CMM

The CMM contains multiple phases where the different parts are performed in. These parts are dependent on each other's input, in order to initially start the follow-up phase. Hence, the phases must be executed in a specific order and it should be guaranteed that the required deliverables are realized.

4.5 CMM linked to EAG

In order to show how our designed CMM is related to EAG, we linked the CMM content towards the core EAG attributes. We grouped the metrics into several topics, and linked these topics with the core EAG attributes in order to show how each topic fosters EAG on detailed level. The concerning tables are provided in Appendix C.

5 Conclusion

Nowadays, technology is involved with almost any change in an organization. Especially in business transformation and -digitalization, new technologies are introduced. These new technologies should be compatible with all current systems and applications of an organization to be able to implement it successfully.

Containers provide organizations with the ability to change their current way of supporting, deploying, and maintaining both hardware and software. Besides that, containers have characteristics that align with the characteristics of EAG, baptizing containers as an 'agile-enabling technology' in our research.

We designed a maturity model for containers which is focused on the use case of SDLC. The CMM consists of differentiating levels of skill regarding the utilization of containers. Organizations who use the CMM, can either derive knowledge on how to start utilizing containers, or use the model to indicate their current performance and state their maturity level, in order to determine from the defined metric specifications how to further improve their SDLC configuration. The corresponding PDD provides an overview of the model's sequence, and supports future research by clarifying what deliverables are realized by which phase and activity.

Finally, before containers can be used to their full potential, they require certain characteristics in the architecture, applications, and provisioning technique. A MSA pattern supports these container requisites, as this pattern decouples applications and systems into small, independent services or individual pieces of software. This makes the applications and services short-lived, stateless, and lightweight. Hence, the combination of the mean of containers, the fundament of the MSA style adopted in an EA, enables IaC and lets the organization use the CMM to further increase the maturity of their SDLC process. The higher the maturity, the easier it gets to advance in enterprise agility.

6 Discussion and future work

Despite the success of containers, the technology also comes with several downsides. Containers' major weakness is the lack of isolation – due to their shared kernel with other containers – restricting containers from being used in cases where security is essential. To fix this issue, two main options are available:

- Further research on containers to improve isolation;
- Hybrid architecture solution of putting one or more containers inside a VM.

New OS images have been developed to improve performance of VMs. This resulted in the concept of *unikernels*, which can be seen as a 'lean' OS image where only the essential parts that are required by a desired application are implemented. Another concept is the m*odular and stateless* OS image. Using such a unikernel or modular OS

image for a VM instead of a full OS image is likely to save computing resources and increase speed. Combined with a container, these new architectural concepts can make use of both the benefits of containers, and the isolation functionality of VMs. We think it is interesting to conduct further research on how the popular hybrid solution can be further advanced using different technologies.

The current version of the CMM does contain untapped potential that requires additional research. CICD components of each SDLC phase have different interrelationships with each other. We think that having certain CICD components enabled in a SDLC configuration, this also enables other components to exist more easily or improve their functionality. Hence, realizing untapped potential (i.e. 'quick wins') an organization may possess. Knowing those underlying cross-links of the metrics, enables researchers to further develop the CMM, by using the interrelations to ultimately provide a more accurate, realistic, and organization-specific advice in the Results matrix.

For defining the numbers of all values used in the Business Area, we had to base our model on the given prices by providers of such services, and several articles. However, the formulas could be extended, as current calculations are limited in regards to the real word. It is likely that organizations (B2B) can receive discount or other payment constructions that were not visible during our research period. Future research could focus on deriving more realistic values from relevant organizations, in order to improve the validity of the Business Area and bring the results closer to the real-world context.

Finally, we propose to further validate the CMM, and further conduct research to measure and test the CMM's effect through case studies at different types of organizations. We believe it would be interesting to include both larger and more advanced organizations in the case studies, as well as smaller organization who specialize in software development, and middle-sized traditional organizations who may be ready and interested in starting to apply containers.

References

[1] Overby, E., Bharadwaj, A. & Sambamurthy, V. Eur J Inf Syst (2006) Enterprise Agility and the enabling role of information technology. 15: 120. https://doi.org/10.1057/palgrave.ejis.3000600.

[2] Kaddoumi, T., & Watfa, M. (2016, August). A proposed agile enterprise architecture framework. In Innovative Computing Technology (INTECH), 2016 Sixth International Conference on (pp. 52-57). IEEE.

[3] Sherehiy, B., Karwowski, W., & Layer, J. K. (2007). A review of enterprise agility: Concepts, frameworks, and attributes. International Journal of industrial ergonomics, 37(5), 445-460.Dawson, P., & Bittman, T. J. (2008). Virtualization changes virtually everything. Gartner Special Report.

[4] Oracle. (2012). Oracle® VM. 1.1.1. Brief History of Virtualization. Retrieved from https://docs.oracle.com/cd/E26996_01/E18549/html/VMUSG1010.html

[5] Portworx. (2017). Portworx Annual Container Adoption Survey 2017.

[6] Buckley, K. (2017, January 10). Application containers will be a $2.7bn market by 2020, representing a small but high-growth segment of the Cloud-Enabling Technologies market. Retrieved from https://451research.com/blog/1351-application-containers-will-be-a-$2-7bn-market-by-2020,-representing-a-small-but-high-growth-segment-of-the-cloud-enabling-technologies-market.

[7] Yusuf, Y., Sarhadi, M., Gunasekaran, A., 1999. Agile manufacturing: the drivers, concepts and attributes. International Journal of Production Economics 62 (1–2), 33–43

[8] Tseng, Y.-H., & Lin, C.-T. (2011). Enhancing enterprise agility by deploying agile drivers, capabilities and providers. Information Sciences, 181(17), 3693–3708. http://doi.org/https://doi.org/10.1016/j.ins.2011.04.034

[9] Pal, N., & Pantaleo, D. (Eds.). (2005). The agile enterprise: Reinventing your organization for success in an on-demand world. Springer Science & Business Media.

[10] Sharp, J. M., Irani, Z., & Desai, S. (1999). Working towards agile manufacturing in the UK industry. International Journal of production economics, 62(1-2), 155-169.

[11] Lee, J., Siau, K., & Hong, S. (2003). Enterprise Integration with ERP and EAI. Communications of the ACM, 46(2), 54-60.

[12] Holbeche, L. (2015). The Agile Organization: How to build an innovative, sustainable and resilient business. Kogan Page Publishers.

[13] Rigby, D. K., Sutherland, J., & Takeuchi, H. (2016). Embracing agile. Harvard Business Review, 94(5), 40-50.

[14] Dove, R. (1999). Knowledge management, response ability, and the agile enterprise. Journal of knowledge management, 3(1), 18-35.

[15] Lewis, J., & Fowler, M. (2014). Microservices: a definition of this new architectural term. Mars.

[16] Wieringa, R.J. (2014) Design Science Methodology for Information Systems and Software Engineering. Springer Verlag

[17] Dreyfus, S. E. (2004). The five-stage model of adult skill acquisition. Bulletin of science, technology & society, 24(3), 177-181.

[18] Team, C. P. (2002). Capability maturity model® integration (CMMI SM), version 1.1. CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI-SE/SW/IPPD/SS, V1. 1).