

## MASTER

### Proposing a method for governing federated development teams under a low-code paradigm

Slangen, C.A.M.

*Award date:*  
2021

[Link to publication](#)

#### **Disclaimer**

This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

#### **General rights**

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

Eindhoven University of Technology

# Master Thesis

Proposing a method for governing federated development teams  
under a low-code paradigm



**Author**  
*C.A.M. Slangen*

**Research question** – ‘How should enterprises govern federated development teams under a low-code paradigm?’

By

C.A.M. Slangen

*Amsterdam, 30/04/2021*

**Institute** *Eindhoven University of Technology*  
**Master faculty** *Industrial Engineering & Innovation Sciences*  
**Master track** *Innovation Management*  
**Group** *Information Systems*

**Committee members**

dr. Maryam Razavian	<i>First supervisor   Eindhoven University of Technology – Information Systems Group</i>
dr. Baris Ozkan	<i>Second Supervisor   Eindhoven University of Technology – Information Systems Group</i>
dr. Laura Genga	<i>Third Supervisor   Eindhoven University of Technology – Information Systems Group</i>
Daan van Dillewijn	<i>Company supervisor</i>

*'The coronavirus is gripping our country. Us and the rest of the world. Together we are faced with an enormous task.'*

*M. Rutte (2020)*





# Abstract

In today's rapidly changing business environment, enterprises are required to adapt to changes quickly. Low-code software development aims to enable this by offering easy to understand platforms, on which applications can be built relatively quickly, without the need for full-stack developers to create complicated and time-consuming software. Such low-code platforms are characterised by their ability to be organised and scaled in a federated way, where the central entity coordinates and governs the autonomous federated teams. This research examines what the most effective governance model is for low-code development teams, from the perspective of the central entity. The study is performed by conducting semi-structured interviews, with a broad set of interviewees from different industries and disciplines. A proposed governance model is subsequently presented, which is composed of results from a literature study and explorative interviews. Based on the obtained data, a process, operating model, and four mechanisms are proposed that support this model. The effectiveness of the governance model is tested in a case company, through which its mechanisms are evaluated on their utility, quality, and efficacy during two confirmatory evaluation focus groups. The development of a security self-assessment mechanism, an explanation and transparency of the development process and the creation of an internal community are found to support governing federated development teams. Furthermore, introducing an intake process is found to lead to increased central monitoring capabilities and central development demand.

**Keywords:** *Low-code development, governance models, federated structure, digital transformation, autonomous software development teams, centre of excellence*

# Preface



This thesis marks the last phase of the time I spent in my life as an official student. During this period, I had the privilege to have an extraordinary time. Remarkably, this last phase consisted of a turbulent time since this thesis was written during the COVID-19 pandemic. One iconic quote during the Dutch prime minister's first press conference marked the start of this time. Despite the quote not being admirable on its own, it reflects upon the past months' processes and challenges when writing this thesis. I would not have thought the pandemic would affect me as severely as it did. Both mentally and personally, probably as many others, I have gone through an intense year with numerous obstacles; there have been many setbacks during this journey. I have experienced it as a challenging, long-lasting, yet instructive period.

When reflecting, it feels like a blur where many emotions and thoughts have passed. I still met many new people online, learned about an emerging and exciting technology, and managed to drag myself out of a slump on multiple occasions. Moreover, as icing on the cake, I managed to land my first job.

To be able to achieve this and have the perseverance, numerous people supported me. Therefore, I would like to express my gratitude to some. First, I would like to thank my mentor, Maryam, for having regular meetings where we discussed my thesis, situation and process. It was helpful how you showed your understanding and were compassionate; this allowed me to be open and honest, and it helped me continue with my research. I would also like to thank Baris for his helpful feedback and quick responses, especially in the last phase. Next, I would like to thank my colleagues at the case company. First, Lucy, for granting me the opportunity to start my internship, even in these unstable times. I would not have been able to get this interested in the technologies and governance concerns without this opportunity. It was inspiring to review and discuss the governmental issues from a managerial perspective. Next, my colleagues from the low-code team, especially Daan, with whom I had meetings almost every day. You were always open to explain me the low-code technology and answer my questions. Moreover, other colleagues helped me and were open to having interviews or evaluations, like the one-on-one sessions with Katrien and Chris. Following, I would like to thank all interviewees for making time to conduct an interview with me. Because of you, this research is noteworthy since it researches governance from multiple perspectives. Lastly, I would thank my friends and family who supported me along the way and motivated me.

Best regards,

Christophe Slangen

*Amsterdam, April 2021*



# Executive summary

## 0.1 Introduction and problem identification

Taking the perspective of the central entity, this report researched how enterprises should govern federated low-code development teams. In this context, a federated structure is an organisational structure where autonomous subunits are integrated via one central entity. The research was initiated within the central low-code department of a case company. Here, a federated structure was implemented to foster local development teams that will build applications with a low-code application platform. In this federated structure, the central team acts as an overarching role, and the federated teams will have a level of autonomy to build applications. While this federated structure has numerous benefits, risks and limitations are also involved in implementing such a governance structure in a low-code development context, which is mainly driven by the fact that the autonomy also means that the quality of the federated teams' developments cannot be fully guaranteed. Moreover, quality issues and risks range from security and compliance risks to scalability limitations. To help examine these issues and potential solutions, the following research objective was formulated:

---

*Improve the collaboration between federated teams and a central team by setting up a governance model such that the federated low-code teams understand what to do in order to establish a more secure and reusable development process where manual work is reduced*

---

## 0.2 Research design

For this study, the design science research methodology by Peffers et al. (2007) is applied. The research starts by defining the problem and showing its importance for the research. Thereafter, a deeper understanding of the objectives and improvements of the to-be designed artefacts are researched. Then, a development process, a corresponding federated operating model and four mechanism artefacts are developed and demonstrated in a case company. The artefacts will be evaluated, and its results are communicated to discuss the practical and theoretical implications. Below, the research question is presented followed by the corresponding sub-research questions:

***'How should enterprises govern federated development teams under a low-code paradigm?'***

- **SRQ1** *What characterises the low-code paradigm?*
- **SRQ2** *What characterises federated governance, and how can federated software development be governed?*
- **SRQ3** *How does the low-code paradigm relate to federated development?*
- **SRQ4** *What mechanisms, tasks and best practices can be used to govern federated development teams under a low-code paradigm?*

The research questions were answered using various methods, including a documentary technique, a literature study, and semi-structured interviews, which were conducted with a set of interviewees from different industries and functions.

## 0.3 Literature study and explorative case study

A thorough study of the academic literature on the subject presented an answer for the first sub-research question, as multiple low-code characteristics were found that define the low-code paradigm. These characteristics were (i) *Ease of use*, (ii) *Types of users*, (iii) *Reusability*, (iv) *Development method*, (v)

Easy management and monitoring, (vi) Collaboration, (vii) Interoperability with other systems, (viii) Scalability, (ix) Security, and (x) Speed of development. Moreover, five risks and limitations were found: (i) Scalability, (ii) Type of users, (iii) Extensibility, (iv) Interoperability among LCAP, and (v) Compliance and security.

The second research question explored how central teams in a similar structure should govern federated software development teams. As this research looked at governance from a central perspective, eight tasks for this central team were found from the literature study. These tasks were used as input for the coding process of the last sub-research question.

The third sub-research question analysed how the two concepts, the low-code paradigm and federated governance, relate to each other and explored what benefits and problems arise while combining both concepts. Finding these relationships is deemed essential to understanding the issues that occur in low-code operating models. After a coding process, five strengths were found; (i) transparency in applications, (ii) more local developers, (iii) monitoring and access capabilities, (iv) quality assurance, and (v) quick local development. In addition to these benefits, three risks were identified: (i) higher risks to compliance and security issues, (ii) the risk that the federated teams do not adapt or change to a new way of working, and (iii) the risk for lack of expertise. These risks are taken into account when developing the artefact.

The fourth sub-research question identified best practices, mechanisms and tasks for central teams in federated low-code governance models. The output that was generated in sub-research question 2 was used as input for the coding process. However, before defining the mechanisms and tasks, it is found that these teams' governance depends on four factors; (i) Type of technology, (ii) Type of organization, (iii) Type of application, and (iv) Maturity of the team, which can be seen in Figure 1.

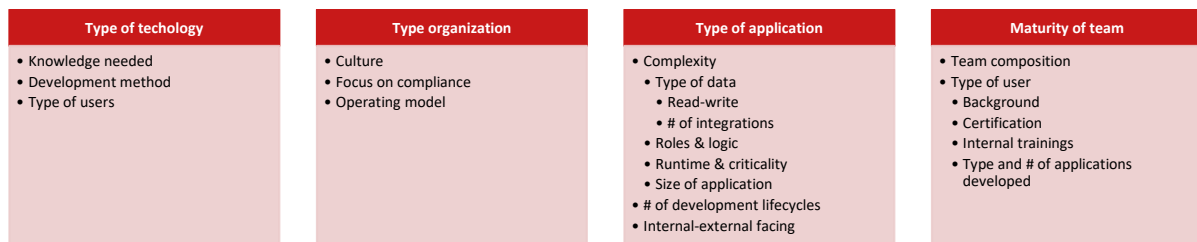


Figure 1: Dependencies for a federated low-code governance model

The influence that central teams should have should be more strict in, for example, complex and external-facing applications or when a team has just started working together on a certain project or task. Therefore, this study focused on finding a governance model for starting federated teams. In addition, it is advised to first start with developing less complicated applications.

Based on the conducted explorative interviews, a set of eight categories of tasks was identified that central teams should conduct in a federated low-code context to ensure that teams work efficiently and deliver quality applications while maintaining a level of control from the central team. The degree of execution and emphasis placed on these tasks depends on the factors described in Figure 1. The tasks can be found in Figure 2 and are elaborated on in subsection 4.3.3.




















Figure 2: Central tasks for governing low-code federated development teams



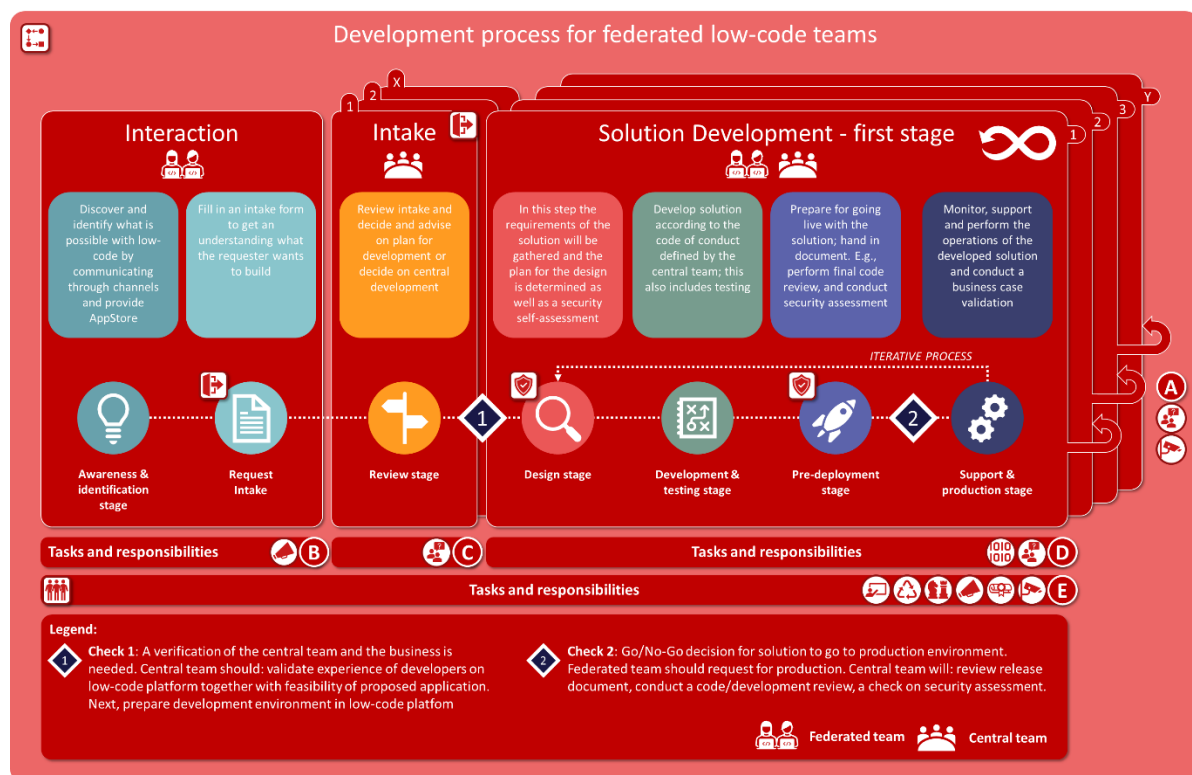
## 0.4 Artefact design

Based on the data obtained, a development process and operating model were proposed to govern federated low-code development teams. Moreover, four mechanisms were developed that support this model and process to manage the teams. The defined process and operating model can be found in *Figure 3* and *Figure 4*, respectively. In both these figures, the designed mechanism artefacts are indicated through square icons, as indicated in *Table 1*.

**Table 1:** Selected mechanisms for this research to be designed

Mechanism artefact	Description	Covering tasks and responsibilities
	Building an internal community Create a community with regular meetings for presentations and one channel where the teams come together	  
	Develop the intake process Develop a process for federated teams and other colleagues to submit their ideas to understand the application to be built and advise and assist them on the development	 
	Develop security self-assessment Develop a security self-assessment that allows the federated team to assess the security level of the application and to alert them on the measures that should be taken accordingly	  
	Explain and provide transparency on the development process Define a process ( <i>Figure 3</i> ) and create an internal online channel with explanations on this complementary to the existing manual and create explanatory information for the federated teams	    

\* Pre-defined stakeholder goals (SG) and requirements (Req.) from *Table 3*



**Figure 3:** Solution development process for federated low-code teams

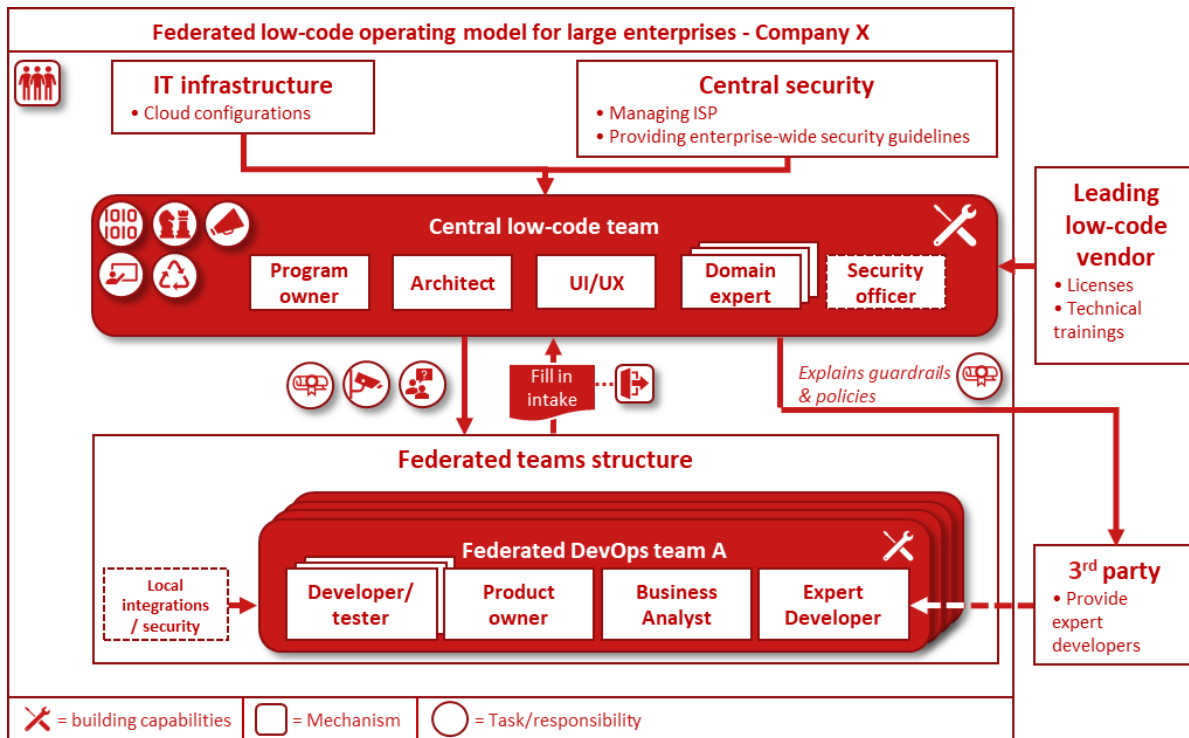


Figure 4: Operating model federated low-code model

This research studied what mechanisms could be used to support the governance of federated teams that would lead to the desired behaviour. Therefore, the developed mechanism artefacts ensured and contribute to improving the governance of the federated teams.

## 0.5 Artefact evaluation

The mechanism artefacts were evaluated with stakeholders from a case company to review the usability, quality and effects in a real context. From the evaluations, it appeared that the explanations and transparency on the development process were helpful in combination with the developers manual. Moreover, an internal community ensures that federated teams are inspired and trained on executing the proper techniques and mitigating the barrier between the central and federated teams. However, it did appear that such an internal community is more useful as the number of federated teams increases. Next, the security self-assessment mechanism helped explain the security guidelines for federated teams and the central security officer's audit tasks. With this mechanism, federated teams can better understand better what the requirements are from the beginning of development. Lastly, the intake process appeared to be less effective for understanding projects, as a meeting with the requester was still preferred. However, an intake process did seem to give more attention to the technologies and promoted the central team in general within the company. This can lead to a higher adoption rate of federated teams in the long run.

## 0.6 Conclusion & recommendations

Since the artefacts outputs may be biased, as it only focuses on a particular type of company, the proposed governance model and the process should be interpreted and used as a starting point for other organisations that want to implement federated low-code teams. As the most effective way to govern teams depends on several factors, central teams' main task should be to understand the developments of the federated teams and their maturity level. In this way, governance and corresponding processes can be adapted to mitigate the dysfunctional barrier between the central and federated teams and focus on the design freedom within the federated teams.

In addition, because of the dependencies, the focus should shift from governing federated teams to the process of governing them by assessing which processes should be in place to ensure design freedom while maintaining quality within the developments. Creating a zoning model and categorising which applications can and should be built and by whom, could help make sure that the right people will develop the applications that need more supervision. Moreover, having internally open and accessible explanation documents on the method of working, next to a Single Point of Contact that is also using it, would reduce work for the central team. Furthermore, a security self-assessment mechanism improved the knowledge of federated teams and their autonomy and reduced manual work for the central team. Moreover, an internal community would ensure that federated teams are aligned and get inspired, which led to increased reusability. Additionally, an intake process increased the demand for projects although it did not improve the understandability of applications of federated teams. Likewise, central teams should additionally give internal training, in which the codes of conduct, company security guidelines, and the development way of working are explained. Based on certificates of the technology, internal training, and other factors, federated teams should be able to earn more rights and design freedom. In this way, central teams could emphasise autonomy where possible and prevent becoming a bottleneck. Therefore, the trust that is obtained by the federated teams could be expressed in a set of factors. These factors could be used to adjust the control mechanisms and development process per use case to enhance design freedom and maintain control and quality.



# Table of contents

<b>ABSTRACT</b>	<b>I</b>
<b>PREFACE</b>	<b>II</b>
<b>EXECUTIVE SUMMARY</b>	<b>III</b>
<b>0.1 INTRODUCTION AND PROBLEM IDENTIFICATION</b>	<b>III</b>
<b>0.2 RESEARCH DESIGN</b>	<b>III</b>
<b>0.3 LITERATURE STUDY AND EXPLORATIVE CASE STUDY</b>	<b>III</b>
<b>0.4 ARTEFACT DESIGN</b>	<b>V</b>
<b>0.5 ARTEFACT EVALUATION</b>	<b>VI</b>
<b>0.6 CONCLUSION &amp; RECOMMENDATIONS</b>	<b>VI</b>
<b>TABLE OF CONTENTS</b>	<b>VIII</b>
<b>LIST OF ABBREVIATIONS</b>	<b>X</b>
<b>1. INTRODUCTION</b>	<b>1</b>
<b>1.1 PROBLEM IDENTIFICATION</b>	<b>1</b>
<b>1.2 CASE COMPANY DESCRIPTION</b>	<b>3</b>
<b>1.3 RESEARCH OBJECTIVE</b>	<b>5</b>
<b>2. RESEARCH DESIGN</b>	<b>7</b>
<b>2.1 RESEARCH DESIGN &amp; METHOD</b>	<b>7</b>
<b>2.2 DATA COLLECTION METHODS</b>	<b>8</b>
<b>2.3 ROADMAP FOR THIS STUDY</b>	<b>14</b>
<b>3. LITERATURE STUDY</b>	<b>15</b>
<b>3.1 LITERATURE STUDY APPROACH</b>	<b>15</b>
<b>3.2 THE LOW-CODE PARADIGM</b>	<b>15</b>
<b>3.3 FEDERATED STRUCTURES AND FEDERATED SOFTWARE DEVELOPMENT GOVERNANCE</b>	<b>21</b>
<b>4. EXPLORATIVE CASE STUDY</b>	<b>31</b>
<b>4.1 INTERVIEW SETUP</b>	<b>31</b>
<b>4.2 RELATION LOW-CODE PARADIGM TO FEDERATED GOVERNANCE</b>	<b>32</b>
<b>4.3 MECHANISMS, TASKS AND BEST PRACTICES OF FEDERATED LOW-CODE DEVELOPMENT TEAMS</b>	<b>38</b>
<b>5. ARTEFACT DESIGN</b>	<b>50</b>
<b>5.1 ARTEFACTS OVERVIEW</b>	<b>50</b>
<b>5.2 PROBLEM UNDERSTANDING CONTEXT</b>	<b>51</b>
<b>5.3 SELECTING MECHANISMS</b>	<b>51</b>
<b>5.4 ARTEFACTS DESIGNS</b>	<b>54</b>
<b>6. DEMONSTRATION &amp; EVALUATION</b>	<b>61</b>
<b>6.1 ARTEFACT DEMONSTRATION</b>	<b>61</b>
<b>6.2 ARTEFACTS EVALUATIONS</b>	<b>62</b>
<b>6.3 EVALUATION CONCLUSIONS</b>	<b>66</b>
<b>7. DISCUSSION AND CONCLUSION</b>	<b>68</b>
<b>7.1 DISCUSSION AND CONCLUSION</b>	<b>68</b>
<b>7.2 THEORETICAL IMPLICATIONS</b>	<b>71</b>

<b>7.3 PRACTICAL IMPLICATIONS</b>	<b>72</b>
<b>7.4 LIMITATIONS</b>	<b>72</b>
<b>7.5 FUTURE RESEARCH</b>	<b>73</b>
<b>8. REFERENCES</b>	<b>76</b>
<hr/>	
<b>9. APPENDIXES</b>	<b>81</b>
<b>APPENDIX I</b>	<b>81</b>
<b>APPENDIX II</b>	<b>82</b>
<b>APPENDIX III</b>	<b>83</b>
<b>APPENDIX IV</b>	<b>86</b>
<b>APPENDIX V</b>	<b>87</b>
<b>APPENDIX VI</b>	<b>88</b>
<b>APPENDIX VII</b>	<b>92</b>
<b>APPENDIX VIII</b>	<b>93</b>
<b>APPENDIX IIX</b>	<b>94</b>
<b>APPENDIX IX</b>	<b>98</b>
<b>APPENDIX X</b>	<b>109</b>



# List of abbreviations

Explanations of abbreviations and commonly used definitions in this study can be found in *Table 2*.

**Table 2:** List of abbreviations

Abbreviation	Description
• <b>Citizen developers</b>	Single developers within an enterprise that make use of IT platforms to create new business applications or processes
• <b>CoE</b>	Centre of Excellence
• <b>DevOps teams</b>	Development and Operations is a term used in software development where the two disciplines in a development process are part of one team
• <b>Full-stack developer</b>	A developer that has a lot of software development experience, primarily in traditional software programming languages such as .NET or JAVA and has a computer science-related background
• <b>Go-live</b>	In software development, when an application moves from the test to the production environment where it becomes available
• <b>GSD</b>	Global Software Development, distributed development teams that are developing software
• <b>ISP</b>	Information Security Procedure
• <b>LCAP</b>	Low-Code Application Platform, leading platforms with similar characteristics and features
• <b>MDD:</b>	Model-Driven Development, a method where software is designed by using models. This development method is seen as a precursor of low-code
• <b>SDLC:</b>	Software Development Lifecycle, this is the entire development process when creating an application of process
• <b>SPOC</b>	Single Point of Contact, a role that is assigned to a team that manages the contact with that particular team



# 1. Introduction

In this chapter, the origin of the problem and the importance of the research will be explained. Moreover, the case company is briefly introduced in *section 1.2*. Furthermore, the research question and related sub-research questions are presented in *section 1.3*.

## 1.1 Problem identification

Modern times require businesses to respond faster to changes. As shown recently, in the sudden event of a pandemic (see *Appendix V*), this demand for digital adaptiveness is even more highlighted. Low-code is a new type of development that can fit into this demand by enabling new users to quickly build applications using drag-and-drop building blocks (Richardson et al., 2014; Sahay et al., 2020). The platforms offer a new software development method to build complete functional operational applications by making use of easily understandable interfaces that allow to quickly write software code (Colantoni et al., 2020; Sahay et al., 2020; Tisi et al., 2019; Vincent et al., 2020; Waszkowski, 2019). Next to the development speed, many enterprises are introducing the platforms because they are cost-efficient, and a new type of developers can start building applications (Sanchis et al., 2020; Waszkowski, 2019). Furthermore, these platforms are interesting for enterprises since it speeds up development processes without much effort in training, installation and configuration (Waszkowski, 2019). Some even say that by 2023 more than half of the medium to large enterprises will have adopted a low code application platform (Vincent et al., 2020).

However, the implementation of low-code platforms is also associated with risks. For example, it can lead to shadow IT<sup>1</sup> and risks for compliance and security issues since companies do not always test the created solutions (Sanchis et al., 2020). Next, because of the platforms' quick learning curve, inexperienced developers are also using the platforms. These developers could come from the business who have no background in developing software (Colantoni et al., 2020). However, these inexperienced developers could quickly create an application in the wrong way, as shown in *Example 1*.

**Example 1:** Risk of using low code application platforms with inexperienced developers (De Vries, 2019)

---

*Eventhough low code application platforms consist of easy drag and drop interfaces, it is quite easy to create 'spaghetti code', which is a term for unstructured and hard-to-maintain code. On low-code application platforms, lists can be added in applications easily, however, developers should also put a limit to those lists to prevent the application from fetching millions of rows. Thus, setting boundaries is important when developing applications, even on low-code platforms.*

---

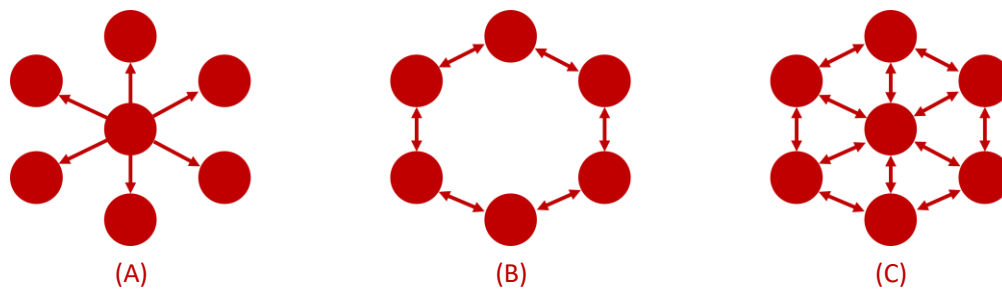
Most of the time, low-code platforms are implemented in companies by starting with a small team (Tiemens & Weel, 2019). When the low-code platform is expanded within a company, the risks as described will become more relevant. Therefore, governance, including policies, ways of working, and codes of conduct, should be implemented to ensure specific quality levels. The governance and the alignment between IT and the business can be set up in various ways and depend on the organisational structure (Lindström et al., 2017; Weill & Ross, 2004a).

Most enterprises are divided into subdivisions, e.g. sales, finance, R&D, marketing. Larger enterprises are even organised into multiple divisions or regions consisting of numerous subdivisions. Organisational structures have much impact on the outcomes that an enterprise generates, and deciding on the type can

---

<sup>1</sup> According to Cisco (2020) 'Shadow IT is the use of IT-related hardware or software by a department or individual without the knowledge of the IT or security group within the organization. It can encompass cloud services, software, and hardware'

be challenging (Lindström et al., 2017). In *Figure 5*, the three primary organisational structures are shown; (A) centralised, (B) decentralised, and (C) federated.



*Figure 5: Three types of organisational structure (Lindström et al., 2017)*

The circles in *Figure 5* can be seen as subdivisions within enterprises. In a centralised structure (A), the power and governance lie in the centre, which flows to the subdivisions (Lindström et al., 2017). This creates a tight structure with strict control and increased coordination and supervision (Mintzberg, 1979). A decentralised structure (B) does not have a central unit, and therefore the decision power and authority are shared, and tasks are conducted separately (Mintzberg, 1979). Lastly, there is a federated structure (C). On a higher abstract level, the concept of federated structures can be found in many contexts. Think of the political domain context; countries like the United States of America or the Soviet Union have these structures. In these cases, the subdivisions have different autonomy and heterogeneity levels (Sheth & Larson, 1990). There is a central point, however, ‘the steering forces between the centre and subunits are bidirectional and exist between the subunits, making the central unit informal’ (Lindström et al., 2017, p. 159). However, this informality differs, and the level of governance and decision-making power the central point should have is a challenge in federated structures and differs per context.

When looking at IT governance, such as low-code, enterprises can use various methods to develop software to stay competitive. One way to achieve this is by implementing, Distributed Software Development (DSD) teams. According to Tufekci et al. (2010, p. 150), ‘[DSD] emerged as a business need in the global world, and had been previously referred to as outsourcing, off-shoring, multi-site development, distributed development, and “Software Engineering over the Internet”’. Nowadays, this is referred to as Global Software Development (GSD). However, this distributed way of developing software also has its challenges and, therefore, the authors propose to have federated software development (Tufekci et al., 2010). In this proposed federated software development, comparable IT departments that develop similar solutions can work autonomously but share information or deliverables with the other teams.

Thus, as suggested, software developments should be done in a federated structure. However, this federated structure also implies that federated teams in a federated system will be autonomous compared to a centralised structure (A) with distributed teams (Roth, 2014; Sheth & Larson, 1990; Tufekci et al., 2010). Because of the autonomous and heterogenous characteristics, federated teams can be different concerning management policies, functional requirements and ways of working. Furthermore, a lack of coordination or governance in federated structures can lead to problems, such as a lack of inefficiency (Williams & Karahanna, 2013). To overcome these problems, explicit coordination, mechanisms or actions should be implemented to improve these problems and overcome obstacles for alignment in federated IT structures (Brown, 1999; Sambamurthy & Zmud, 1999). Lacking or having poor coordination or governance for how teams should collaborate in a federated IT structure could otherwise lead to wasted resources because of duplication, diseconomies of scale, higher costs, and lower productivity (ITGI, 2006; Strassmann, 1995).

In addition to this, in the research of Bourgault, Drouin and Hamel (2008), the authors found that success in managing these type of teams depends on the level of team autonomy and formal decision-making processes; teams should be able to conduct their activities autonomously, but success is increased



when transparent formal decision-making processes are established. This emphasises the need for a formal governance structure with rules, policies, ways of working, and requirements. Similarly, in the research of Asfarmanesh & Camarinha-Matos (1997), the authors suggest enterprises make a ‘special workflow plan’, which can be seen as an explicit plan to explain what is expected from these teams. This special workflow should guide federated teams and their developers to increase quality within their software developments by following the desired behaviour. Moreover, it is essential to specify every federated team’s desired cooperation behaviour (Afsarmanesh & Camarinha-Matos, 1997).

However, it has not yet been investigated what this desired behaviour is in the context of low-code and how enterprises should implement a governance model for low-code federated development teams. In other words, because of the low barrier to implementing low-code platforms within an enterprise (Richardson et al., 2014), combined with the benefits of speeding up software development to enhance digital transformation (Sahay et al., 2020; Sanchis et al., 2020; Waszkowski, 2019), low-code development is getting much attention. Therefore, most enterprises start small with one team or even several citizen developers developing applications with the low-code platform. The next phase is to scale this across enterprises which can be done in a federated way. In this federated low-code structure, there should be transparent governance to ensure this desired behaviour is followed. Therefore, it is argued that it remains unclear in the literature how a central team within a federated structure should set up proper federated governance in the context of low-code, which includes the processes, operating model, and tasks to guide these development teams.

Similar to the research of Williams & Karahanna (2013), this study did not attempt to identify or provide a list of mechanisms for this collaborative process between the central and federated teams. This research sought to understand how enterprises with similar low-code IT issues should set up a governance model to improve governance between the federated teams and the central team. To achieve this, various mechanisms could play a role to support this governance.

## 1.2 Case company description

The problem identification started with a problem analysis at a case company and is a large beverages company located in the Netherlands. Moreover the case study will take place in the global intelligent automation division. The company is currently active on all continents and is, in terms of revenue, one of the leading companies in the world. This study is conducted at the central low-code development team within this global division. A visual overview of this department can be found in *Figure 21* in *Appendix I*.

The team had adopted and maintained the low-code platform OutSystems at the case company. More specifically, the team is the platform owner of OutSystems within the case company. Therefore it took care of all platform related issues. Thus, the team’s task was to make sure the technology platforms on which the solutions were developed were up-to-date, secure and that the solutions and platform were consistent with the codes of conduct. Next to facilitating the OutSystems platform, the team delivered its products in a centralised way. Therefore, the team consisted of developers that could build custom-made applications on request. An example of how this was done can be seen in *Example 2*. This example showed that the central team was responsible for development, operations (running of application or process) and support of the developed solution.

### *Example 2: Way of working centralised model*

---

*When division A wants to create an application to solve a problem within their business, they will get into contact with the central low-code team. Then, the central low-code team develops the solution with the low-code platform to solve the problem. When the software development is done, division A will be charged for the development costs and support and the solution will be deployed. When adjustments have to be made, division A will request changes and this process will be executed again.*

---

### 1.2.1 Problem orientation

Because of the high number of regions and divisions worldwide and each region wanted to digitise, the central team received many requests for building applications. Since the central team had to focus on both creating applications and maintaining the platform, the team would not be able to conform to the demand. Therefore, the case company wanted to introduce a federated model where the delivery model will change to a federated structure with federated teams established in regions or divisions.

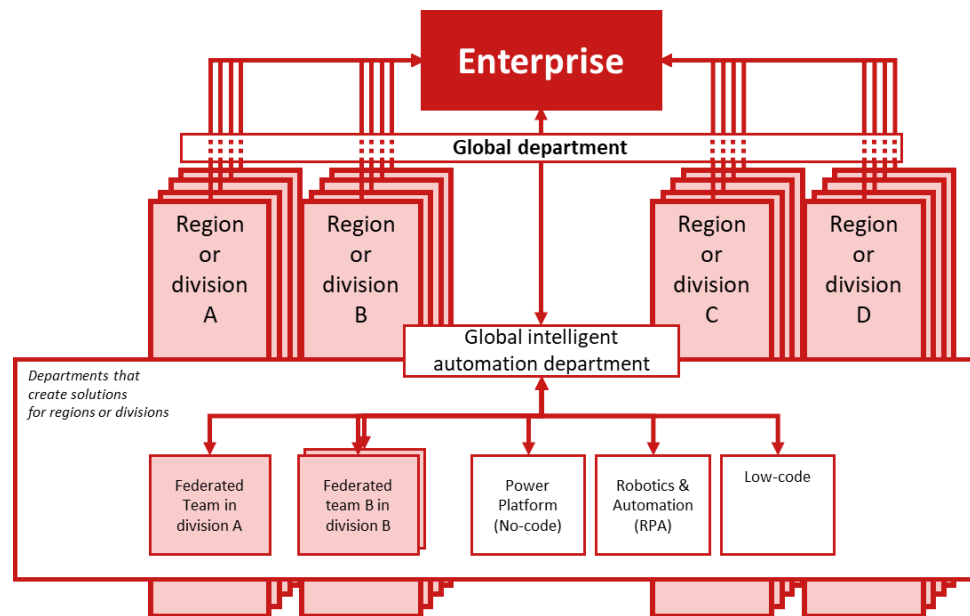


Figure 6: Federated model for developing business-related software applications

Figure 6 shows how this new federated model looked like. In this new architecture, regions or divisions could choose to create solutions in a centralised or federated way. Both the federated teams and the central teams just had one general goal: solve business problems by developing solutions. As shown in Figure 6, regions or divisions do not have to establish federated teams; they can also still use the building capabilities of the developers in the central teams who will develop solutions for them in the central model. Therefore, when the central model was chosen, applications would be built according to the previously discussed method, and the central team was fully responsible. However, when the federated model was chosen, divisions and regions could establish their own federated team and build solutions on top of the three central team' technology platform. In addition, in Figure 6, there were two federated teams shown in the region or division B. This implied that it was also possible for an division or region to establish multiple federated teams that could focus, for example, on a different type of applications.

One of the case company's long-term goals was to become the most digital beverages company in the world and to be more cost-efficient. Therefore, the transition to a federated architecture, where this study is a part of, contributed to the case company's higher goal. In a federated architecture, federated teams should be governed and guided to ensure security and to be able to benefit from sharing knowledge between federated teams (Sanchis et al., 2020). Before this study, it was unclear how the central team should govern the federated teams as a central entity for the case company to make sure the federated developments were executed in the right way. To scope this research, it was assumed that the low-code platform already gained popularity within the company. More specifically, the enterprise already believed in the capabilities low-code offers and wanted to expand the capability across the company via a federated structure.

### 1.3 Research objective

The main question regarding the federated model for the case company was that it is unknown how to establish this collaboration in the federated model. More specifically, when the teams developed applications centrally, they would understand how applications should be developed. However, when establishing federated teams, the teams were autonomous and, therefore, they could decide on what and how they wanted to build their solutions. Consequently, the central team could not ensure that the federated teams would develop solutions in the same way. The federated model would be most beneficial if the federated teams were aligned and delivered applications of a high level of quality. When working more closely together, knowledge sharing may be more applicable. This also allows enterprises, for example, to scale up developments by reusing developed solutions since developed products can be shared. In addition, from the first explorative interviews, it became clear that the federated teams will also consist of team members that do not have a development background. Since the central teams were the federated teams' platform owners, they wanted to ensure that security, code quality, and reusability could be guaranteed. This implied that the central teams should find a method to influence the federated teams on the 'how part' of developing solutions, especially when there was a possibility that teams could consist of non-experienced developers. Therefore, the following problem statement was formed:

---

*'It remains unclear how enterprises should govern federated development teams under a low-code paradigm. Currently, the right collaboration between the central teams and these federated teams is not yet defined resulting in a risky, unautomated, and unscalable situation where federated teams do not know what to do'*

---

The research problem was divided into several components, which can be seen in *Table 3*. From the research, an artefact was designed that solved the problem. For the artefact, the list of requirements on the artefact level for the case company can be found in *Appendix IV*, in *Table 21*. This list was iteratively developed which implies that requirements were added and modified during the research process.

**Table 3:** Design problem

Design Problem	
<b>Problem Context</b>	A lack of a process and structure in which the proper way of working of federated low-code development teams are defined to govern these teams
<b>Stakeholder goals</b>	<p><b>A:</b> Find a method that guides federated teams to improve software developments within federated teams</p> <p><b>B:</b> Define a governance structure that improves the collaboration between federated teams and the central team to increase software development quality within these teams.</p> <p><b>C:</b> Gain insight into how this federated structure supports the design for reusability among federated teams</p> <p><b>D:</b> Federated teams should be guided in a way that they understand what to do at what stage</p> <p><b>E:</b> It should fit in the current context and way of working</p>
<b>Artefact requirements</b>	<p><b>F:</b> It should help or support for a design for security</p> <p><b>G:</b> It should help or support for a design for reusability</p> <p><b>H:</b> It should be designed in a way that manual work is reduced</p>
<b>Artefact</b>	A method that defines the governance between central and federated teams that submits to the artefact requirements (guidance/method/process/review tool/structure)

As can be seen from *Table 3*, the artefact requirements differed from the stakeholder goals. The artefact requirements would drive the artefact's design, whereas the stakeholder goals were the ones that could be achieved by implementing the artefact. This gave the following design objective:

---

*Improve the collaboration between federated teams and a central team by setting up a governance model such that the federated low-code teams understand what to do in order to establish a more secure and reusable development process where manual work is reduced*

---

### 1.3.1 Research questions

This research focused on designing an artefact that enabled central teams to improve the governance between the central and federated teams by proposing a governance model and providing insights into best practices and mechanisms. Since this study sought to contribute to the literature, a research question was defined on an abstract level that would cover other industries and the case company:

---

***'How should enterprises govern federated development teams under a low-code paradigm?'***

---

Four sub-research questions were formulated and were answered to provide a clear answer to the main question. The first three subquestions were related to the problem investigation phase where the objectives and fundamentals for the artefacts were formed. The first sub-research question identified the aspects introduced by the low-code paradigm by looking at the characteristics. New elements will be presented in this low-code paradigm compared to 'traditional' development teams. This question aims to study what characteristics drive the low-code paradigm and are unique to this development method.

**SRQ1.** *What characterises the low-code paradigm?*

The second sub-research question was related to the federated way of working part of the research question. The sub-research question tried to answer what federated governance is and how federated software development teams could be governed from a central entity. By answering this question, a view on federated governance was provided with corresponding tasks that would lead to effective governance.

**SRQ2.** *What characterises federated governance, and how can federated software development be governed?*

The third sub-research question brought the two concepts of federated governance and the low-code paradigm together. With this question, the understanding of the relationship between the two concepts has been researched. It tried to find the risks as well as the benefits of this combination.

**SRQ3.** *How does the low-code paradigm relate to federated development?*

The last sub-research question was related to how enterprises could set up current federated low-code models, what governance mechanisms could be used and how this collaboration between federated teams and central teams should be established. Based on the identified risks and limitations of the previous sub-research, question mechanisms were identified which mitigate this. Furthermore, this question aimed to study how this governance should be established, including research on existing governance models with processes, procedures, and policies and understand best practices.

**SRQ4.** *What mechanisms, tasks and best practices can be used to govern federated development teams under a low-code paradigm?*

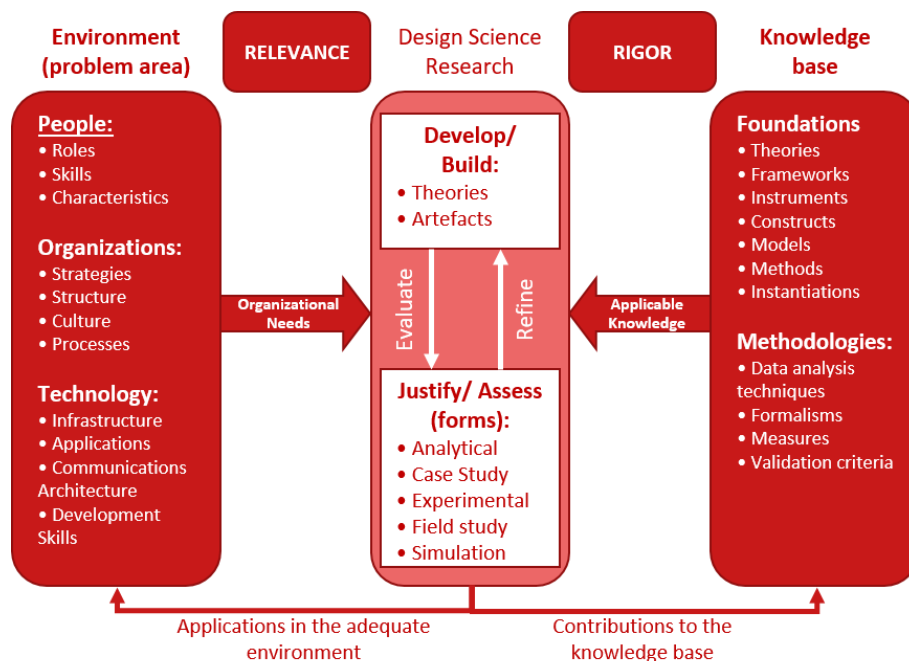


## 2. Research design

In this chapter, the applied research methods are discussed in *section 2.1*. Moreover, the data collection methods and the rationale for choosing these methods are described in *section 2.2*. Lastly, the roadmap of this study including the stages of the used research methodology, and the data collection methods are described and visualised in *section 2.3*.

### 2.1 Research Design & Method

This study applied a design science research methodology (DSRM) because its origin starts with a business problem. According to Hevner et al. (2004), design science is focused on solving problems by creating an innovative purposeful artefact for a stated problem or within a problem domain. This artefact should be purposeful and, therefore, it should solve the stated problem (Hevner et al., 2004). Moreover, the artefact should be a technology-based solution that is linked to a fundamental and relevant business problem (Hevner et al., 2004). Hevner et al. (2004) designed a model that showed how relevance and rigour play a role in design science research, shown in *Figure 7*. For this research, the artefacts were evaluated within the problem area, and after evaluation, the contributions were added to the scientific knowledge base. The artefact was evaluated and refined where experimental methods were used during the development.



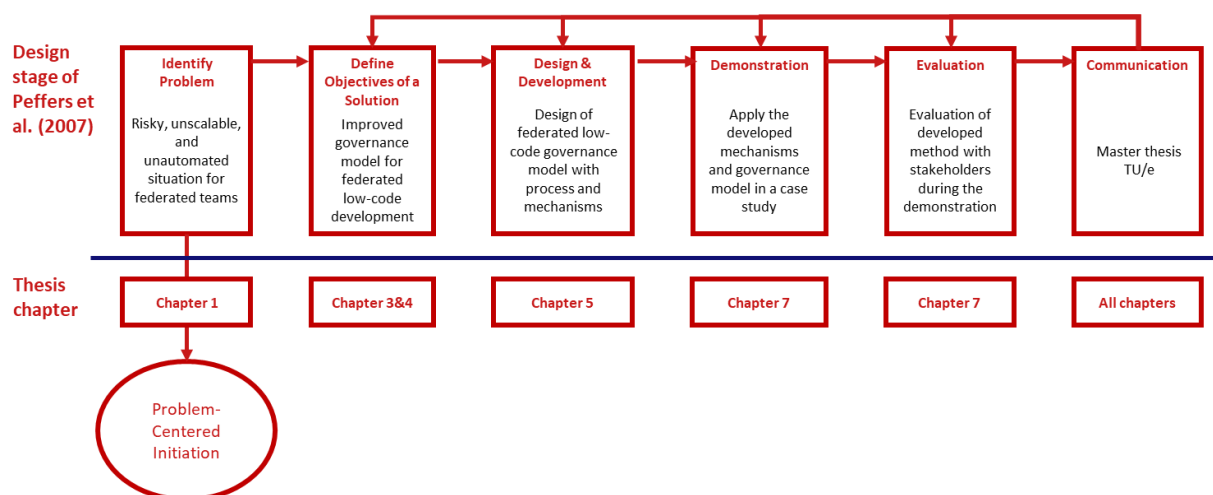
**Figure 7:** Relevance and rigour in design science research by Hevner et al.(2004)

As shown in *Figure 7*, artefacts should be built based on the organisational needs (e.g. input and requirements of the organisation and stakeholder goals) and by applying the information found in the knowledge base. This knowledge could also be contextual knowledge gathered from persons in the environment (Dresch et al., 2015).

More recently, Peffers et al. (2007) created a new design science research methodology for research that used a structure based on the study of Hevner et al. (2004). In this research, the authors 'represents a

unique effort to formally define a research methodology for use in [Information Systems] IS' (Peffer et al., 2007, p. 73). The presented model is divided into six steps and can be seen in *Figure 8*, and the original model can be found in *Figure 31*. Since this study was started from an observation, the process began with a problem centred initiation approach. Therefore it started with activity one of the DSRM model provided by Peffer (2007) and can be followed in the sequence, as is shown in *Figure 8*.

When looking at the design stages of Peffer et al. (2007), identifying the problem is explained in *sections 1.1 and 1.2*. Next, the objectives of the solution should be defined. Based on the formulated research questions in *subsection 1.3.1*, sub-research questions were answered based on a literature study. The way how this method was used is described in *section 2.2*. After this, a general understanding of federated governance and the low-code paradigm was established. Through an explorative case study as described in *Interviews*, the last two sub-research questions (**SRQ3 & SRQ4**) were answered to find ways and best practices to govern low-code federated teams. Based on this knowledge base, an operating model, development process could be defined, and a collection of mechanisms was provided. From this list, a subset was selected that were designed. This artefact design happened in the 'design and development stage' of Peffer et al. (2007). The designed artefact's contribution could be embedded in the artefact itself (Peffer et al., 2007). Then, the solution was demonstrated in a case study in a business context which is in line with the demonstration stage of the DSRM model of Peffer et al. (2007). After the demonstration, the artefacts were validated with the main stakeholders using confirmatory focus groups (see *Focus groups*). The chapters of the study are linked to the design stages of the selected research method of Peffer et al. (2007) and can be found in *Figure 8*.



**Figure 8:** DSRM Process Model by Peffer specified to the current research (Peffer et al., 2007)

## 2.2 Data collection methods

Dresch et al. (2015) explained a way how to conduct scientific research in design science. In this book, the authors presented six techniques for gathering information are provided. To answer this study's sub-research questions, a majority of the methods was used; a documentary technique, a literature study (bibliographic), interviews, and focus groups. Moreover, to make sure that the artefacts are applicable in the case context, interviews will be held within the company. Furthermore, interviews with experts outside the organisations were also conducted to obtain general information on topics that could be applied to this research. The data collection methods used per sub-research question are shown in *Table 4*. Below *Table 4*, descriptions of the used data collection methods are presented.

**Table 4:** Data collection methods per research

SRQ	Research question	Data collection methods
1.	What characterises the low-code paradigm?	<ul style="list-style-type: none"> <li>• Documentary</li> <li>• Literature study</li> </ul>
2	What characterises federated governance, and how can federated software development be governed?	<ul style="list-style-type: none"> <li>• Documentary</li> <li>• Literature study</li> </ul>
3	How does the low-code paradigm relate to federated development?	<ul style="list-style-type: none"> <li>• Qualitative semi-structured interviews</li> </ul>
4	What mechanisms and tasks can be used to govern federated development under a low-code paradigm?	<ul style="list-style-type: none"> <li>• Qualitative semi-structured interviews</li> </ul>
	Evaluation	<ul style="list-style-type: none"> <li>• Confirmatory focus group</li> </ul>

### 1. *Documentary technique*

A documentary technique allows a researcher to study and assemble previous information on a research topic and is, most of the time, the first step in gathering information (Saunders et al., 2012). This method includes analysing documents that can be textual or non-textual such as pictures, audio or video recordings (Dresch et al., 2015). For this study, internal processes, technology architectures, and technology descriptions were analysed to grasp the business problem's contextual barriers and the current way of working.

### 2. *Literature study (bibliographic)*

This technique was used to collect previously developed information on a particular theme to discover a new topic. In this way, the researcher could investigate a matter that has been studied in a new light. For this technique, the research could use, e.g. books, scientific papers or conference proceedings (Dresch et al., 2015). All these papers were stored in Mendeley, a software tool that allows users to manage scientific documents efficiently. In this research, scientific digital databases, such as Google Scholar, JSTOR and IEEEExplore, were predominantly used. These databases offered a broad set of relevant papers for this study. Moreover, tools to find new articles such as TU/e Library search and Mendeley's recommendations were used to discover further related information. In addition, books were used to have a more coherent view on Design Science Methodology and IT governance. These were either purchased during the research traject or before this study and used during the courses of the master.

For the literature study, several techniques were used. For example, the snowballing method was used to find related articles. The snowball method can be used forward and backwards and works by looking into references that are cited in articles or searching articles that cited a specific article (van Aken et al., 2012). This method helped to find a large share of scientific papers on particular topics (van Aken et al., 2012).

For the low-code paradigm, mainly recent articles were used since it could be seen as an emerging technology (Sanchis et al., 2020). However, the term low-code could be traced back to older terms in software engineering such as Model-Driven Development (MDD), Rapid Application Development or even end-user design (Cabot, 2020; G. Fischer et al., 2004; Lefort & Costa, 2019; Waszkowski, 2019). Since low-code development is growing, including these low-code platforms' features and capabilities, there was less focus on older terms of low-code. To answer **SRQ2**, a literature study was conducted focusing on the tasks that should be performed within central teams in a similar software development context. For answering this question, a comparison has been made with centres of excellence, which is explained in *subsection 3.3.3*.

### 3. *Interviews*

For answering **SRQ3** and **SRQ4**, a broad explorative case study is conducted where interviews are used as a data collection method. According to Diccio-Bloom et al. (2006), interviews can be classified into three classes: unstructured, semi-structured and structured. First, unstructured interviews can be seen as guided

conversations. Examples are when the investigator observes a group and picks various candidates while observing to ask questions (DiCicco-Bloom & Crabtree, 2006). Secondly, structured interviews follow a script of pre-defined questions and only asks these. The interview technique can obtain information that cannot or are harder to get with the bibliographic approach. In this way, the interviewer can obtain specific information related to the context of a case. However, according to Saunders et al. (2012), this technique also has downsides, e.g. there always is a communication barrier between the interviewer and the interviewee. This barrier includes ways interviewees can interpret questions and problems that the interviewer cannot avoid, such as the possibility of holding back information. Lastly, semi-structured interviews allow the interviewee to adapt and change questions in situations where the researcher sees a fit for the interviewee to understand the question better and for the interviewer to get a better idea of the data collected (Saunders et al., 2012). Moreover, this method has added advantage over quantitative methods by providing more explanatory information, which can refine this study's proposition (Shull et al., 2008). Moreover, because the conversations remain open and flexible, it is also more accessible for the interviewer to receive more detailed questions (DiCicco-Bloom & Crabtree, 2006). Therefore, for this explorative case study, semi-structured interviews were used to make sure the right questions could be formulated beforehand and topics could be explained and verified when something is unclear during the interview

Therefore, through semi-structured interviews, the two sub-research questions **SRQ3** and **SRQ4** were answered. During the interviews, the sequence of the questions asked may vary across all interviews; this allows for a more natural conversation (Knox & Burkard, 2009). However, a 'natural conversation' was not possible since all interviews were conducted virtually via Microsoft Teams due to the COVID-19 pandemic (see *Appendix V*). Therefore, physical interpretation during a conversation was more complicated, and there was limited time available since interviewees had another meeting planned directly after the interview. Nevertheless, this made it easier to conduct interviews with people that would otherwise have no time available since it lowers the interview barrier because an online meeting could be arranged easier.

Two types of interviews were conducted. First, the interviews with various low-code platforms and a low-code consultant were conducted to answer **SRQ3**; *'How does the low-code paradigm relate to federated development?'* These interviews were performed to understand the relation between low-code and federated governance. Therefore, it was researched how this low-code paradigm would also change how these federated teams should be governed. Moreover, the list of low-code characteristics found in **SRQ1** was reviewed to validate the low-code paradigm definition. In this way, low-code development and a federated governance model was researched using literature research, and the combination was explored with experts in practice during semi-structured interviews.

It was decided to interview a heterogeneous set of low-code platform vendors and neutral technology consultants to get a broad and most objective view as possible. When having this relationship in place, it was analysed how a central team can govern the federated low-code development teams. This brings us to the second interview conducted with various big enterprises and consultants that already implemented a federated structure with software development platforms or advised on these operating models. The interviews with other industries were used to do a cross-case analysis that examined what similarities and differences exist between enterprises whereby each insight is compared with each other (Shull et al., 2008; van Aken et al., 2012). The consultants were used to derive best practices, mechanisms and ways to design a low-code governance model. The input from the first interviewees was also used for this.

For both interviews, predominantly employees were interviewed with much experience within the company or the role and who could examine 'governance' from a managerial perspective. Therefore, these roles could provide more holistic and grounded answers based on experience and knowledge. Furthermore, it was assumed that the interviewees already understood the low-code paradigm's characteristics because of their experience. Therefore, during an interview, the stakeholders had a similar level of understanding. To make sure that the interviewee and interviewer were aligned on the federated



structure, an explanation of the definition of a federated model was provided before the interview itself. Moreover, the interview protocols containing the questions were sent beforehand to all interviewees to prepare for the interview questions, which can be found in *Appendix IIX* (Knox & Burkard, 2009).

All the formal semi-structured interviews that were answering the sub-research questions were video recorded<sup>2</sup>. After each interview, the interviews were transcribed into text and processed via NVivo software<sup>3</sup>. This software helped the researcher to categorise data and find links between them (Shull et al., 2008). Both the transcriptions, recordings and codes were reviewed iteratively. In this process, it was essential to consider the context of each passage of the interview that was grouped to get a constant comparison (Shull et al., 2008). This method was used since it aimed to answer why and how questions (Shull et al., 2008). In the context of this research, these questions were; how should federated low-code be governed, how is low-code related to a federated way of working, and what mechanisms and tasks can help to improve this.

The coding process used the grounded theory as defined by Miles & Huberman (1994). A constant comparison method was used for answering **SRQ3** since they were first open-coded, later axial coded and then selective coded (Shull et al., 2008). This is the process of reassembling the data that was captured and broken apart during the open coding (Shull et al., 2008). In the open coding process, labels or tags were attached to chunks of text which were later categorised and refined (Shull et al., 2008). These codes were post-formed codes which means that they were created during the analysis and coding process (Shull et al., 2008).

However, for answering **SRQ4**, a list of categories of tasks of a central team in software developments was already established in **SRQ2**. This list was later used during the axial and selective coding process to answer **SRQ4**. This can be seen as a template approach. A template approach allows the researcher to code with a set of coding categories based on prior research (DiCicco-Bloom & Crabtree, 2006; Tremblay et al., 2010). Therefore, for identifying the tasks for central teams, preformed codes were used during the coding process that were already defined (Shull et al., 2008). This pre-formed list of codes was useful for the researcher since it allows to holistically approach the interviews when having this knowledge in place. The interviewees with the other industries were not used during the coding process for **SRQ4** because they would have a subjective view compared to the low-code vendors and consultants.

Furthermore, for the explorative cross-case analysis, a visual model of each interviewee's governance structure was created to identify similarities and differences. This method was used since visual models or maps could help when the interviews' data is exploratory (Shull et al., 2008). Graphical models were also beneficial because they take up less space and allowed the researcher to easily depict insights (Shull et al., 2008).

#### 4. Focus groups

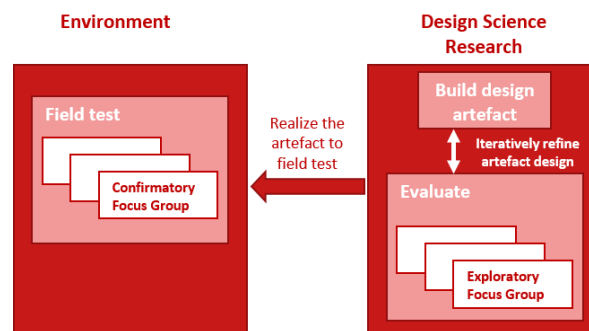
According to Saunders et al., *'focus groups can be perceived as an in-depth interview that occurs in groups with structured sessions that contemplate the proposal, the size, the components, and the procedure for conducting the group'* (Saunders et al., 2012). In the process of design science research, the evaluation of a created artefact is a crucial part. One way of doing this by using the focus group technique. According to Tremblay et al. (2010), this technique can be split into two types; exploratory and confirmatory focus groups. First, to explore and evaluate the context or problem, which will enable the researcher to refine an artefact. Secondly, it allows the researcher to review the developed artefact in a confirmatory focus group in practice. This is visualised in *Figure 9*. In the context of this research, both types were used to evaluate the developed artefact. When the artefacts were created, they were evaluated and validated using confirmatory focus groups. A confirmatory focus group tries to find the designed artefact's utility

<sup>2</sup> Before each interview session, permission is asked due to confidentiality and personal related issues as defined in the General Data Protection Regulation (GDPR).

<sup>3</sup> A commonly used software program to conduct qualitative data analysis

(Tremblay et al., 2010). In this research, a focus group within the case company will evaluate the designed artefact.

Focus groups are a commonly used research method and provide insights into opinions and similarities on a particular topic among group members (van Aken et al., 2012). Therefore, the focus groups were focussing on different designed mechanisms. Next to utility, quality and efficacy must be examined rigorously (Hevner et al., 2004). In addition, neutral questions were asked during the focus group to avoid bias (Shull et al., 2008). Similar to the interviews, due to the pandemic situation (see *Appendix V*), the focus groups were conducted online via Microsoft Teams. Furthermore, for the pre-defined requirements (shown in *Table 3*), additional questions were asked to evaluate the utility, quality and efficacy. The process of how the focus groups are conducted can be found in *Table 5*.



*Figure 9:* Focus group in design research from (Tremblay et al., 2010, p. 603)

*Table 5:* Process focus group method

Part	Researcher tasks
<b>Pre-focus group</b>	
1	Gather participants and prepare evaluation questions (Tremblay et al., 2010)
<b>Focus group</b>	
2	Provide an introduction, introduce the goals and purpose of the session. Provide details on which topics are going to be discussed
3	Elaborate on the artefact and start a discussion about the fulfilment of artefact requirements
4	Start a discussion on utility, efficacy, quality, and stakeholders' goals
<b>Post-focus group</b>	
5	Analyse and interpret data & report results (Tremblay et al., 2010)

### 2.2.1 Data quality

Validity, reliability and controllability should be taken into account to ensure quality research (van Aken et al., 2012; Yin, 2013). Validity is divided into three categories; internal, external, and construct validity (van Aken et al., 2012). These were all taken into account in this research and are shown and described in *Table 6*.

*Table 6: Empirical quality in this study*

Category	Definition	Approaches in study
<b>Controllability</b>	To be able to understand how the research is executed (van Aken et al., 2012)	Providing transparency in scientific methods and approaches used Presenting initial and final coding schemes and interview protocol available (see <i>Appendix IX</i> and <i>Appendix IIX</i> )
<b>Reliability</b>	Whether the outcome of a study will be similar if it would be replicated by another researcher (Yin, 2013)	A broad set of interviewees from different companies is selected for the study Triangulation is used; using multiple techniques to obtain data (Yin, 2013). See <i>Figure 10</i> and <i>Table 4</i> . Having regular meetings with the company supervisor Reviewing results with the company supervisor
<b>Validity</b>	Research is valid when it is justified by the way it is generated (van Aken et al., 2012)	
<b>Internal validity</b>	When a causal explanation is sought instead of a correlational one (Yin, 2013)	Saturation of data is tried to achieve
<b>External validity</b>	The generalizability and transferability of the research (van Aken et al., 2012)	Interviewing multiple people from different companies and industries Having different type of interviewees with other intentions
<b>Construct validity</b>	Establishment of the correct operational measures for the constructs being studied (Yin, 2013)	Triangulation is used; using multiple techniques to obtain data (Yin, 2013). See <i>Figure 10</i> and <i>Table 4</i>

This research is conducted to be as valid and reliable as possible. However, there could still be a sub-continuous researcher bias that could influence how the respondent is answering (Shull et al., 2008). Though, since predominantly experienced respondents were selected, this influence of the researcher is mitigated.

### 2.3 Roadmap for this study

This research is set up according to a pre-defined process shown in *Figure 10*. First, a problem was identified within the industry and verified if there was a gap in the literature regarding this problem. Then, this problem was analysed using literature research and interviews with experts. A list of outputs were generated from this research: recommendations and best practices on how federated low-code teams should be governed, what tasks and responsibilities the central entity has, and defining a process and governance operating model. Subsequently, from this output, a list of mechanisms was formulated, and a development process and operating model were proposed. From this set, multiple mechanisms were chosen that were designed to support the proposed operating model and process. Then, these mechanisms were implemented and demonstrated in a case study. Lastly, the developed tools were evaluated, validated, and the results were documented.

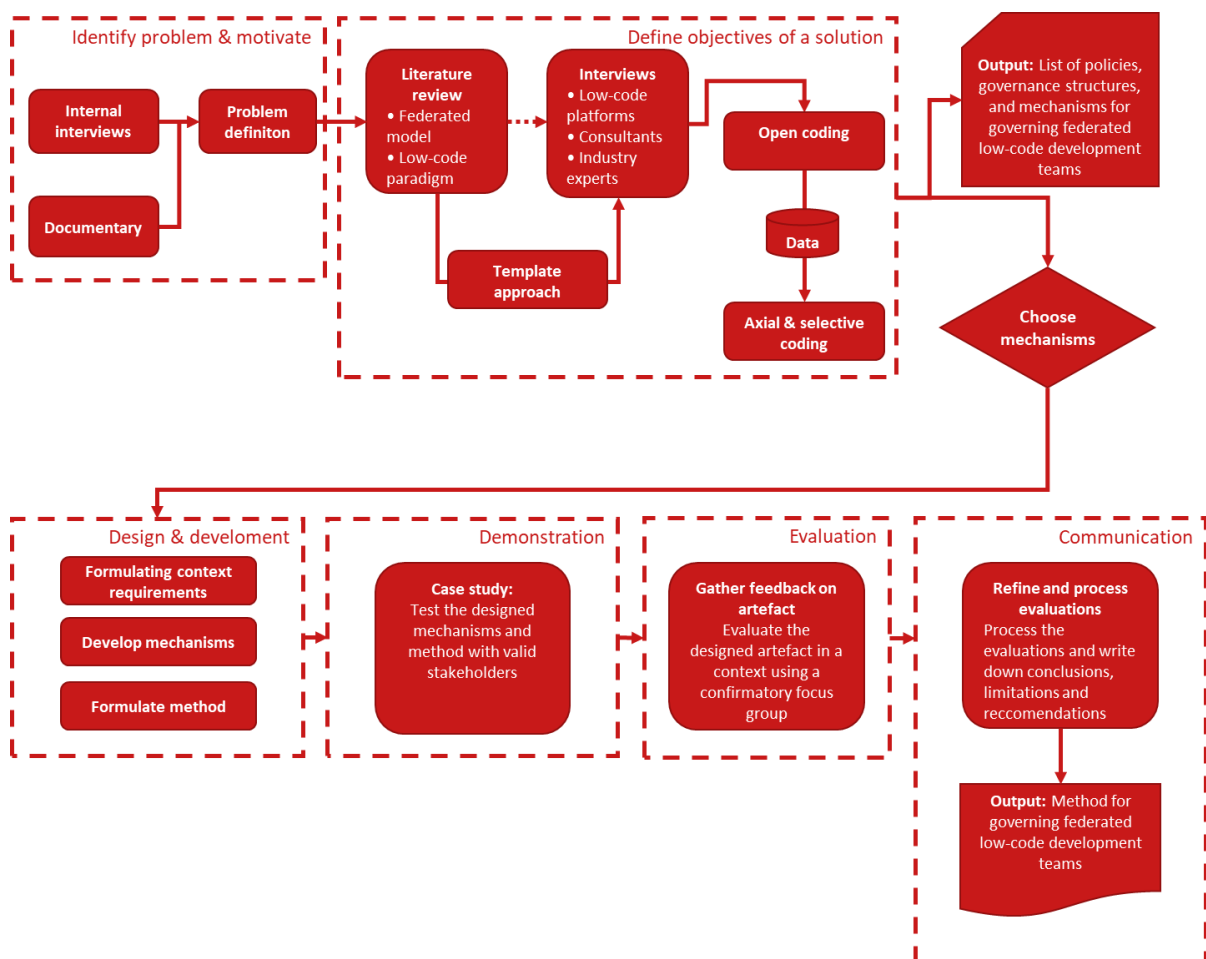


Figure 10: Process of the study according to the research methodology of Peffers et al. (2007)

# 3. Literature study



This chapter will clarify the concepts that are discussed in this research. The chapter starts by defining the low-code paradigm and what this new technology characterises. Then, federated structures and federated governance are explained. After this, tasks for federated software developments are analysed. In this way, the research problem is diagnosed holistically since it first diverges by exploring and analysing the topics and finding new perspectives. Therefore, this research starts analysing on a high abstraction level, looking at the concepts introduced like the low-code paradigm, federated structures and IT governance. The literature research continues by converging, synthesising, consolidating topics to study. This chapter will answer the following research questions:

- **SRQ1** 'What characterises the low-code paradigm?'
- **SRQ2** 'What characterises federated governance, and how can federated software development be governed?'

## 3.1 Literature study approach

Two sub-research questions are answered in this section. Both sub-research questions found an answer in another discipline. For each discipline, different search terms were used to find related articles. The most used search terms are included in *Table 7* to produce a compact and straightforward table. However, slight variations of these terms were also used.

Table 7: Search terms

Nr.	Search term	Source
<b>Low-code</b>		
1.	Low-code development	(Lefort & Costa, 2019; Richardson et al., 2014; Sahay et al., 2020; Sanchis et al., 2020; Sattar, 2018; Tisi et al., 2019)
2.	High-Productivity Application PaaS	(Lefort & Costa, 2019)
3.	Rapid Application Development (RAD)	(Ruparelia, 2010)
4.	Model-Driven Development (MDD)	(Altintas et al., 2007; Cabot, 2020; Di Rocco et al., 2015; Sahay et al., 2020)
5.	End-user design	(G. Fischer et al., 2004)
<b>Federated governance</b>		
6.	Federated Architecture/ Development/ governance	(Stephen J. Andriole, 2012; Garita et al., 2001; Khosroshahi et al., 2015; Kirschner & Roth, 2014)
7.	IT Governance	(Farwick et al., 2011; Linthicum, 2009; Sambamurthy & Zmud, 1999; Weill & Ross, 2005, 2004a; Williams & Karahanna, 2013)
8.	Autonomous/virtual teams	(Bourgault et al., 2008; Landy & Conte, 2013)
9.	Centre of Excellence/Expertise/ Community of Practice	(Frost et al., 2002; Gray, 2004; Marciniak, 2012; Wenger, 1998)
10.	Global Software Development teams/Distributed Development teams	(Stephen J. Andriole, 2012; Tufekci et al., 2010)
11.	Digital/software factory	(Altintas et al., 2007; Sanchis et al., 2020)

## 3.2 The low-code paradigm

As briefly introduced, low-code development is a relatively new concept, and therefore, not many papers are written on this topic. Consequently, we can conclude that low-code development is an emerging technology (Sanchis et al., 2020). Moreover, the features of low-code platforms vary. Therefore, this new way of working with low-code platforms introduces a low-code paradigm (Sanchis et al., 2020). Thus, low-code platforms present new perspectives to the research of governing federated software teams.

The service of low-code platforms, used in enterprises, can be seen as software as a service (SaaS) and as a platform as a service (PaaS) since enterprises can create software applications with this software. Therefore, variations were made from these traditional service models, such as an application platform as a service (aPaaS). According to Gartner (2020, p. 1): '[aPaaS] is a cloud service that offers development and deployment environments for application services'. Therefore, low-code technology platforms can be seen as aPaaS since they offer various environments to create new applications. However, low-code technology platforms can run both in the cloud and on-premise. This means that these low-code technology platforms were falling through the cracks. Therefore, Gartner introduced a new segment called high-productive application Platform as a Service (hpaPaaS) (Alexander, 2019; Gibbons, 1994). According to Gartner's famous hypecycle in 2018, hpaPaaS are plotted in the slope of enlightenment. In this phase, the benefits and opportunities of technology are being understood, and companies are investing in pilots with the relevant technology. Later, this term was changed by Gartner into even a more specific segment which is called Low-Code Applications Platform (LCAP) (Vincent et al., 2019). Though various names can be used for federated development of low-code, in this study, LCAP is used.

As defined by Gartner: '*an LCAP is characterised by its use of model-driven or visual development paradigms supported by expression languages and possibly scripting to address use cases such as citizen development, business unit IT, enterprise business processes, composable applications and even SaaS applications*' (Vincent et al., 2020, p. 1). However, various platforms are available on the market, which differ considerably in the platforms' possibilities. In the yearly report of Forrester, the authors compare these different platforms (Rymer & Koplowitz, 2019). According to Sanchis et al. (2020), one of the low-code limitations is the fragmentation across all low-code vendors. Therefore, depending on each vendor, different low-code paradigms can be defined (Sanchis et al., 2020). To overcome this problem, this research only looks at the low-code platforms labelled as leaders in the yearly Forrester report and Gartner's annual review (Rymer & Koplowitz, 2019; Vincent et al., 2020). These leading low-code application platforms (e.g. Salesforce app cloud, OutSystems, and Mendix) are used to build similar optimal and scalable applications for enterprises (Sahay et al., 2020).

To define the low-code paradigm, characteristics have to be identified that are part of these LCAP. Therefore, recent studies on low-code are analysed to determine the low-code paradigm and have a holistic view. From these papers, the most important and most common characteristics are selected. The following sections will elaborate on the low-code paradigm's main characteristics based on these articles.

### 1. *Ease of use*

As explained in *section 1.1*, LCAP make use of drag and drop interfaces that allow users to build applications with prebuilt building blocks (Colantoni et al., 2020; Sahay et al., 2020; Tisi et al., 2019; Vincent et al., 2020; Waszkowski, 2019). Next to this, some vendors allow users to build applications based on business process diagrams or workflows (Waszkowski, 2019). Because of this easily understandable software development method, there is a steep learning curve where users can quickly learn how to develop within the platform. Moreover, LCAP can drastically reduce learning costs while having large scopes on the possibilities in development options (G. Fischer et al., 2004). Furthermore, because the complexity of development is reduced, it enables a developer to focus more on the end-users requirements (Sanchis et al., 2020). Because the LCAP are also easy to change, it is easier to maintain applications or quickly edit or add features (Sanchis et al., 2020). This alignment between business requirements is in line with the DevOps way of working of low-code (Colantoni et al., 2020).

### 2. *Types of users*

Because of this new and easy way of developing software, lower-skilled computer scientists can start developing applications in these platforms after some short training (Sahay et al., 2020; Sanchis et al., 2020; Tisi et al., 2019). This allows enterprises to train employees from the business to develop software. Therefore, new developers can start building full functional applications after several months of training because of the steep learning curve (Sanchis et al., 2020). Because there is less dependency on enterprises

to attract full-stack developers with lots of coding experience, costs can be saved when building applications by minimising the number of costly expert developers. Furthermore, full-stack developers have more time to focus on the more complex aspects of software development, such as integrations, data infrastructures and managing integrity (Sahay et al., 2020). Therefore, a larger group within enterprises can automate processes and build applications that increase development speed and the number of software developments that can be realised.

### 3. Reusability

To simplify software development, LCAP allow users to develop with prebuilt building blocks from the platform itself or components that are already built. Moreover, connectors or REST-APIs can be reused within the platform. In addition, when developing in the same platform, the developed solutions or applications will be built on top of the same code (e.g. CSS or C+), supporting standardisation. Therefore, similar to other software tools, complete applications can be reused or copied on the application level. However, in some LCAP, references to existing applications or modules can be created. Furthermore, UI/UX templates of applications can be created and reused to maintain a company's style and to ensure a similar customer experience across applications. However, some argue that the discovery and possibilities of reusing assets within LCAP is not yet where it could be (Tisi et al., 2019). Barriers to this reusability arise because, in some cases, the code needs too much modification to reuse it, incomprehensive code or low-code quality (Sanchis et al., 2020; Wu et al., 2019).

### 4. Development method

Product development is about delivering added value to the end customer. Before, in traditional coding, the software development lifecycle (SDLC) was always done in a waterfall method (Ruparelia, 2010). In this method, it was the goal to identify what was needed in terms of requirements and architecture and then deliver the best possible product. When the product was finished, it was checked with the end-user to test and review the developed software. After this, the service and support teams were involved. Low-code software development can be seen as a new synonym for MDD, which can also be traced back to Model-Driven Engineering (MDE) (Cabot, 2020). LCAP can be seen as the modern version of MDE (Bernaschina et al., 2018; Sahay et al., 2020; Tisi et al., 2019). This indicates that LCAP is focussing on modelling software applications rather than coding them. Colantoni et al. (2020) discussed the combination of development & operations (DevOps) with MDE. DevOps defines a group or teams consisting of two functions; development people and people who take care of the operational part (Colantoni et al., 2020). It intends to shorten the SDLC to provide deliverables and features continuously, and it is also part of the agile methodology (Hoda, 2019). An agile way of working can be seen as an enabler of DevOps, whereas any form of cloud computing, such as low-code, can be seen as an enabler of DevOps (Jabbari et al., 2016). This DevOps way of working is often used in low-code because of the development speed when combining the two functions (Sahay et al., 2020; Tiemens & Weel, 2019; Vincent et al., 2020). Because of the ease of use and low-code development speed, minimum viable products (MVP) can be developed relatively quickly (Sanchis et al., 2020). Therefore, it becomes easy to validate developed features with business which shortens the feedback loops and reduces wasted time on unnecessary developments on specified requirements or features (Sanchis et al., 2020). This iterative way of working characterises DevOps. Furthermore, next to DevOps, most LCAP support other collaborative development methods like Scrum and Kanban to ensure that the developers can work efficiently and link their tasks to active sprints (Sahay et al., 2020).

### 5. Easy management and monitoring

As explained, most low-code leading platforms are offered as PaaS solutions where the developed applications are stored in cloud environments (Sanchis et al., 2020). Usually, there are four different environments; development, testing, acceptance and production (DTAP). Each environment has its functionality which is described in *Example 3*.

**Example 3:** Developing an application in the DTAP environments

---

*Imagine a team is developing an application and the team wants to build in a new feature to their application. The developers will start writing the code in the platform in the development environment. In the test environment, the new features of the applications will be tested by the developers. Next, in the acceptance environment, the features will be tested with end users that are going to use the application in a secured environment and when the tests are validated and checked. Then, the new version of the application can be deployed to the production environment where the application runs.*

---

These environments are embedded into LCAP in the leading low-code platforms and are already set up in a way that is easy to control. For example, OutSystems is using Lifetime to manage the applications through the DTAP pipeline. Because of these pipelines, it becomes easier to continuously implement and automatically deploy new features by updating applications to more recent cloud versions (Tjoa et al., 2018). Automating these pipelines can be achieved by implementing continuous integration/continuous deployment (CI/CD) pipelines or a DevOps pipeline (Tjoa et al., 2018). In these pipelines, testing capabilities such as performance tests, security tests and quality tests can be automatically executed before deploying to another environment. In some low-code platforms, similar CI/CD pipeline functions are already incorporated in the platform itself (Sahay et al., 2020).

Next to the entire DTAP pipeline, LCAP provide version control, maintenance and configuration in one single platform (Richardson et al., 2014). In this way, all critical stakeholders such as security officers, architects and business experts can configure and manage the platform (Richardson et al., 2014). Furthermore, in this way, the DevOps way of working can be used more efficiently.

#### **6. Collaboration**

LCAP offer users a collaborative environment that help them to work on applications simultaneously. In addition, developers can share components or knowledge only with people within the company. Moreover, because of the new type of users involved in low-code, most leading LCAP offer huge online communities where (new) users can help each other or share their ideas and applications (Vincent et al., 2020). In these large online communities, developers can ask platform-related questions to anyone on the internet. It offers tutorials on how to build applications, components and displays general information on the platform itself. These large communities stimulate an open-innovation culture by sharing knowledge, parts of code and even applications (Di Rocco et al., 2015). Furthermore, in LCAP, there are easy ways to manage version control within the platform itself (Sahay et al., 2020).

Moreover, because of the DTAP environment in one single platform and the delivery speed in LCAP, apps can be created relatively quickly in sandbox environments, and feedback can be retrieved from customers or employees (Richardson et al., 2014). This collaborative way of working on an application is also related to the previously discussed DevOps method (Colantoni et al., 2020). Therefore, this easy collaboration between developers, users and customers supports the quick feedback loops methodology.

#### **7. Interoperability with other systems**

LCAP allow users to integrate with REST-APIs or connect standard services within large enterprises such as Office 365, Sharepoint, SAP or Dropbox (Sanchis et al., 2020). Because of this feature, LCAP can create complex applications that retrieve information from large external resources. In general, LCAP are also open for cloud integration, integrations with legacy systems, and DevOps toolings such as Docker or Jenkins (Faura, 2019).

#### **8. Scalability**

Some LCAP allow for efficiently scaling applications when the number of users, data traffic, and data storage needs to be increased to work appropriately (Sahay et al., 2020). Moreover, making applications easily scalable is focussed on by low-code vendors since most LCAP vendors design their business model



around fees where organizations have to pay per user and applications in the production environment (Sanchis et al., 2020).

### 9. Security

When developing in LCAP, some level of security is already built-in on the platform level. Furthermore, when looking at the security level of applications developed by LCAP, security features can be easily incorporated (Sahay et al., 2020). Moreover, user access can be managed within the platform and code reviews will scan for security issues (Sahay et al., 2020). However, next to these low-code characteristics, there are also some challenges and barriers in low-code development, such as security and compliance risks (Sanchis et al., 2020), which will be discussed in *subsection 3.2.1*.

### 10. Speed of development

The reusability, ease of use, integration possibilities, and the integrated development tools that are part of LCAP allow for quick development. Therefore, applications can be built with fewer developers in a shorter timeframe (Richardson et al., 2014). The drag-and-drop capabilities and out-of-the-box functionalities of LCAP allow for rapid development because the developer does not have to look at the underlying code (Sahay et al., 2020).

Moreover, because of the new pool of users, enterprises can have more software developers that can build applications simultaneously. In addition, if applications are set up in the right way, they can be scaled easily. Furthermore, because of the used development method, the speed of development is increased even more. This allows developers to build rapidly because they are more close to the business. Because of this speed, applications can be made faster, and the time to market can be increased (Sanchis et al., 2020). Therefore, external applications could start bringing in value quicker, and development costs can be saved.

To conclude, based on the literature and grey literature<sup>4</sup>, low-code development can be categorized into several characteristics. For example, secure environments can be set up, mitigating the risks of developing poor quality applications. Furthermore, because of the platform features, such as the PaaS construction, quick learning curves, and the CI/CD capabilities within the platforms, organizations can minimize the initial costs for setup, which lowers the barrier for implementing such a platform (Richardson et al., 2014). These characteristics could be extended to a longer list, but the most common characteristics are put together to make it more compact. These characteristics that shape the low-code paradigm can be found in *Table 8*.

**Table 8:** Low-code paradigm characteristics based on literature

Nr.	Characteristic	Explanation	Source
1.	Ease of development	Graphical interfaces, pre-built building blocks, drag-and-drop methods and understandable interfaces make it easy to use the platforms	(Sahay et al., 2020; Sanchis et al., 2020; Tisi et al., 2019; Vincent et al., 2020; Waszkowski, 2019)
2.	Type of users	Because of the ease of use of the LCAP and the significant communities that support other users, a new set of developers such as citizen developers can make use of the platform	(Sanchis et al., 2020; Tisi et al., 2019)
3.	Reusability	Within LCAP, (parts of) applications can be reused to enhance the speed of software development	(Sahay et al., 2020; Sanchis et al., 2020; Tiemens et al., 2019)
4.	Development method	LCAP are designed to work in a DevOps way of working accompanied with methods such as scrum and creating MVP's with short feedback loops	(Sahay et al., 2020; Tiemens et al., 2019; Tiemens & Weel, 2019; Tisi et al., 2019)

<sup>4</sup> Grey literature is defined by information obtained from sources such as conference proceedings, seminars that are not commercially controlled and are mainly written for a particular audience (Dresch et al., 2015; van Aken et al., 2012)

5.	Easy management and monitoring of applications	Because of the PaaS construction and CI/CD pipeline functions such as deployment built-in, it is easy to manage the entire SDLC of applications built on the system	(Sanchis et al., 2020; Tiemens et al., 2019; Tisi et al., 2019)
6.	Collaboration	LCAP are designed to work together on projects in real-time and by collaborating via online developers communities. These communities also foster the open-innovation mindset	(Di Rocco et al., 2015; Sahay et al., 2020)
7.	Interoperability with other systems	LCAP provide pre-built-in connectors with large existing systems and allow other platforms to integrate within the system itself seamlessly	(Sahay et al., 2020; Tiemens et al., 2019)
8.	Scalability	The way how applications can be set up in a way that the platform can handle large amounts of users	(Sahay et al., 2020)
9.	Security	A certain level of security is built into the platform and pre-built coding blocks. Therefore a specified security level can be ensured	(Sahay et al., 2020; Tiemens et al., 2019)
10.	Speed of development	With low-code development, applications can be developed at a high pace because characteristics such as pre-built building blocks, collaboration options, interoperability and ease of use	(Sahay et al., 2020; Sanchis et al., 2020)

*Table 8* presents a list, based on literature, that characterizes the low-code paradigm. The four most important characteristics that define the low-code paradigm are its development method, type of users, ease of development, and development speed. These characteristics are interrelated since they influence each other.

### 3.2.1 Risks and limitations of low-code application platforms

The listed characteristics show somewhat positive aspects of low-code development. However, recent research also mentions three main limitations of LCAP: scalability, fragmentation, and the lack of developers' programming knowledge (Tisi et al., 2019)<sup>5</sup>. Other challenges are extensibility limitations and steep learning curves that go hand in hand with users' low programming knowledge (Sahay et al., 2020). In addition, in low-code development, risks with compliance and security might occur as well (Sahay et al., 2020). The following section will elaborate shortly on these risks and limitations.

#### 1. Scalability

It seems strange to see scalability also as one of the challenges within the low-code paradigm. However, according to Tisi et al. (2019), current LCAP are not designed to build large scale applications for many users. Furthermore, some of the LCAP offer on-premise capabilities instead of providing their services on the cloud. However, to make full use of the scalable computational cloud capabilities when applications get more extensive, more complex, or the number of users increases, LCAP should be offered in the cloud (Di Rocco et al., 2015). Furthermore, according to Sahay et al. (2020), it is difficult to evaluate and contribute to the scalability of LCAP due to the lack of transparency of the codes within the platforms.

#### 2. Type of users

According to Sahay et al. (2020), some LCAP might not have intuitive interfaces. Since the platform developers most often lack programming knowledge and there are insufficient learning materials, this could be a risk (Sahay et al., 2020). Therefore, some platforms could be set up more understandable; for example, guiding new developers through the development process by introducing interactive wizards can significantly improve users' development experience (Sanchis et al., 2020).

<sup>5</sup> Lowcomote is an interesting research project funded by the European Union that is 'aiming to train a generation of professionals in the design, development and operation of new LCDPs [or LCAP], overcoming the current limitations, by being scalable (i.e., supporting the development of large-scale applications, and using artefacts coming from a large number of users), open (i.e., based on interoperable and exchangeable programming models and standards), and heterogeneous (i.e., able to integrate with models coming from different engineering disciplines)' (Tisi et al., 2019, p. 2).

### 3. Extensibility

Due to architectural constraints, some limitations do not allow developers to have as much freedom as in traditional software development (Sahay et al., 2020). Nevertheless, many applications that will be built do not need pixel-perfect features, and, therefore, not much developer's freedom is required. In these cases, applications developed with the out-of-the-box functionalities offered by LCAP should be sufficient (Sahay et al., 2020).

### 4. Interoperability among LCAP

Since LCAP want their users to stay within their ecosystem, there is no way to share developed artefacts among LCAP (Sahay et al., 2020). Therefore, there is a need for creating a standardized way that allows for collaborative repositories across LCAP (Di Rocco et al., 2015). Consequently, this is also one of the main goals of the Lowcomote project, which is explained in *Footnote 5*. Other projects, such as vf-OS, builds on an open-code software that allows interconnections between modules in other systems (Sanchis et al., 2020).

### 5. Compliance and security risks

First, since low-code development can quickly provide gain, enterprises can adopt LCAP too quickly with risks for shadow IT (see *Footnote 1*) because the platforms are not officially approved (Sanchis et al., 2020). Furthermore, it can be expected that most building blocks developed by experts or by the LCAP itself are developed securely, and most LCAP provide code quality checks that also cover security aspects. However, all current LCAP do not offer dynamic security analysis that identifies risks when the application is produced (Sanchis et al., 2020). Furthermore, applications still might be developed in the wrong way, as described in *Example 1*.

Lastly, similar to developing other software developments, compliance and security issues should always be taken into account (Sahay et al., 2020). However, due to the development method, this could not be incorporated into the development process. Furthermore, since other types of developers are also going to start developing applications with less development process experiences than professional developers, security issues could not be considered.

## 3.3 Federated structures and federated software development governance

To be sustainable, companies should collaborate and adopt technologies within their environment to keep up the pace. Therefore, a transformation in the structure of how these developments are done can be established. According to Fischer et al. (2007), '*corporate structures comprise organizational structures and processes as well as supporting information systems and technologies*' (p. 14). Moreover, now, more than ever, for example, during the recent COVID-19 pandemic (see *Appendix V*), companies have to adapt to new trends even quicker, and costs should be cut. This highlights the importance of enterprises' strategic decisions as these technology advancements trigger enterprise transformations to change structures in enterprises or their departments.

In the literature, enterprise transformation is concerned with changing enterprise architecture. Enterprise Architecture (EA) can help identify the as-is state, the to-be model, and support this migration or transformation (Chen et al., 2008). Furthermore, EA appears to be effective in bringing business closer to information systems (IS) and IT (Aier et al., 2001; Webster & Watson, 2002). Examples of architectures are centralized, federated or decentralized architectures are discussed in *section 1.1*, with each having its pro's and cons. Since business is changing rapidly, enterprises can move to a federated structure where these autonomous entities build their own solutions to keep up with digitization (Tufekci et al., 2010). Andriole (2014, p. 16) even suggests: '*Federated governance, especially in the era of ready technology, is the only way to exploit operational and strategic ready technology opportunities*'. Nevertheless, before we elaborate on federated governance, we have to understand what federated architectures are.

Federated enterprise architecture is an architecture that promotes collaboration, integration and sharing of information and processes among (semi-)autonomous and decentralized parts of enterprises (Kirschner & Roth, 2014; Roth, 2014). According to Roth (2014) and Sheth & Larson (1990), the entities in federated structures have three main characteristics; distributed, heterogenous and (semi-)autonomous. In these papers, the researchers see federated teams as modelling communities that create their own solutions or information locally similar to autonomous development teams, which are described in the literature as;

*'A specific kind of production team that has control over a variety of its functions, including planning shift operations, allocating work, determining work priorities, performing a variety of actual work task... [autonomous teams] are also known as self-managing or self-directed teams'* (Landy & Conte, 2013, p. 522).

Moreover, according to Roth (2014, p. 42), *'an intact autonomy of components implies that organizational responsibilities for each component can remain unchanged'*. This suggests that federated teams do not influence other federated teams or the system as a whole. This is a critical difference compared to global software development teams that work together.

### 3.3.1 Benefits of federated models

Now, a general idea of federated structures is defined. However, to understand why this is useful and why some even suggest that the only way to exploit opportunities is by having a federated approach to IT (Stephen J. Andriole, 2012), we have to look into this approach's benefits. Federated IT development approaches have various advantages; first, it allows for tight alignment between the development units by sharing information (Williams & Karahanna, 2013). Furthermore, it will enable the centralisation of IT platforms by reducing the IT landscape and using only one platform. More specifically, when a suitable technology platform is selected, the central team can conclude contracts with the corresponding vendor. This contract will be more advantageous since several federated teams can use this simultaneously, and economies of scale<sup>6</sup> can be used (Weill & Ross, 2005). Therefore, license costs per user go down because the number of users goes up.

Another benefit of establishing federated teams is that this decentralization of teams enables enterprises to develop applications in countries with lower labour costs and increase developments in areas with lower IT progress (Tufekci et al., 2010). This enables enterprises to develop IT applications in an efficient way. Furthermore, because teams can be established within regional areas where more local knowledge is available, this decentralization of teams allow for quicker adoption to dynamic environments. Furthermore, it stimulates innovation and creativity (Mintzberg, 1979). Another critical benefit of a federated architecture is that data, information, components or already developed solutions can be shared between federated teams (Roth, 2014). Especially when using a similar technology platform, applications or parts of applications could be reused without much effort (Di Rocco et al., 2015). This enables enterprises to scale up digital developments in an easy way since knowledge is shared between teams. Consequently, the time which is spent on development is saved.

In summary: solutions can be developed simultaneously by numerous teams through minimizing platform cost by closing deals with a few platforms in a centralized way, costs can be saved. Moreover, it triggers countries with lower labour costs to set up software development teams, increasing digitalization in these units. Finally, when using a similar platform, parts of developed solutions, or knowledge that will be developed, can be reused in other parts of the enterprise. In this way, the federated structure has various benefits. However, next to the benefits of federated structures, there are risks involved in a federated structure. First, the biggest and most powerful federated teams get the most attention and have the most influence (Weill & Ross, 2004a). Therefore, next to the difficulties in decision making, some argue

---

<sup>6</sup> Economies of scale are cost advantages that can be achieved when production becomes more efficient and fixed costs can be spread over the number of goods.

that a federated governance model can be seen as the most difficult one (Weill & Ross, 2004a). Hence, governance should be targeted to each level.

Furthermore, there is a dysfunctional effect in federated governance; a barrier between the central entity and the federated teams where there is a misalignment (Brown, 1999). Therefore, one of the main challenges in this federated architecture is how this 'integration' should be established to ensure autonomy across the federated entities while maintaining a synergy (Brown, 1999; Williams & Karahanna, 2013). This integration includes how the governance, responsibilities, and information sharing between the federated architecture teams are worked out and how this central entity should coordinate the federated teams.

### 3.3.2 Governing a federated IT model

To determine how a federated IT governance should overcome this integration barrier, we must define what governance actually is. Governance is a vague term that summarizes the structures, policies, and processes of an organization in which it can monitor performance to ensure that objectives are achieved (Weill & Ross, 2004a). Governance is related to the strategy and desired behaviour, which also covers culture and an organisation's beliefs (Weill & Ross, 2004a). When we look at IT governance, we can define this as:

**Definition 1:** IT Governance by Weill & Ross (2004a)

---

*Specifying the decision rights and accountability framework to encourage desirable behavior in the use of IT – Weill & Ross (2004a)*

---

As shown in *Definition 1*, IT governance defines rights and accountability to ensure that the desired behaviour is achieved. The complexity and the desired behaviours vary in every enterprise (Weill & Ross, 2004a). Therefore, how enterprises should govern IT could be done differently (Weill & Ross, 2005, 2004a, 2004b; Williams & Karahanna, 2013). For example, Ross & Weill (2004a) describe:

*'If desirable behavior involves independent and entrepreneurial business units, IT investment decisions will be primarily with the business unit heads. In contrast, if desirable behavior involves an enterprise-wide view of the customer with a single point of customer contact, a more centralized IT investment governance model works better'* (Weill & Ross, 2004a, p. 8)

These more centralized models or independent and entrepreneurial models can be categorized. According to Weill & Ross (2004a), six archetypes of governance models can be defined; (I) Business monarchy, (II) IT Monarchy, (III) Feudal, (IV) Federal, (V) Duopoly, and (VI) Anarchy (Weill & Ross, 2004a, 2004b). This federated IT governance type can be defined as 'a combination of the corporate centre and the business units with or without IT people involved' (Weill & Ross, 2004a, p. 12).

**Example 4:** Governance models of enterprises vary per decision concept

---

*The IT principles, such as the IT strategy, can be decided by the IT group and the business unit leaders in a duopoly model whereas the IT infrastructure will be defined by the IT specialists in an IT monarchy model and business application needs will be selected in a federated model by the corporate center and the business units (Weill & Ross, 2004a)*

---

Effective governance models can be linked to a different set of concepts. An example of one of the top three governance performers' IT governance structures is shown in *Example 4*. More information on the top performers and the corresponding governance model combinations can be found in *Appendix VII*. As can be seen in *Example 4*, IT governance models can vary per IT decision concept, and each concept consists

of a set of IT decisions that should be taken. A different governance archetype can answer each IT decision concept. These concepts are interrelated and are shown in *Table 9*.

**Table 9:** Important IT Governance concepts by Weill & Ross (2004a)

Nr.	Concept	Definition
1.	IT Principles	Clarifying the business role of IT: High-level statements about how IT is used in the business
2.	IT architecture	Defining integration and standardization requirements: organizing logic for data, applications, and infrastructure captured in a set of policies, relationships, and technical choices to achieve desired business and technical standardization and integration
3.	IT infrastructure	Determining shared and enabling services: centrally coordinated, shared IT services that provide the foundation for the enterprise's IT capability
4.	Business application needs	Specifying the business need for purchased or internally developed IT applications
5.	IT investment and prioritization	Choosing which initiatives to fund and how much to spend: decisions about how much and where to invest in IT, including project approvals and justification techniques

Thus, a promising IT governance model should give answers to all these concepts. Therefore, to establish successful IT governance, Weill & Ross (2004a) describe ten leadership principles that can be found in *Table 10*. These principles should be taken into account when designing effective IT governance for enterprises.

**Table 10:** Leadership Principles of IT governance (Weill & Ross, 2004a)

Nr.	Principle	Description
1.	Actively Design Governance	The process of designing governance is active, and senior managers should be involved as well
2.	Know when to redesign	Governance methods could become less relevant over time or because of other factors. Therefore the governance should change along the way
3.	Involve Senior Managers	Involving senior managers when creating governance structures helps to improve the model because it ensures a synergy across all operations
4.	Make Choices	In IT governance, trade-offs occur. However, conflicting goals that come from not making strategic choices often result in ineffective governance
5.	Clarify the Exception-Handling Process	Exceptions in governance structures that allow business units to deviate from the described rules could sometimes be beneficial for the entire enterprise, but this should be defined.
6.	Provide the Right Incentives	Incentives or rewards can encourage people or teams to follow the desired behaviour
7.	Assign Ownership and Accountability for IT Governance	There should be somebody accountable for the IT governance to make sure governance is implemented in the right way
8.	Design Governance at Multiple Organizational Levels	Dependent on the enterprise's size, governance is designed at each level; organizational-wide, division level, and business unit level.
9.	Provide Transparency and Education	This will help in the confidence of the governance. The less transparent and accessible these models are, the fewer people in the enterprise will follow them.
10.	Implement Common Mechanisms Across the Six Key Assets	Implementing mechanisms that support the proposed IT governance structure should be similar across all enterprise-critical assets.

Especially number 8, design governance at multiple organizational levels, is vital for this research since we will look at federated governance on a platform level. Therefore, we have to zoom in on a federated approach on this level.

### 3.3.3 Centre of Excellence

In the first instance, when looking at implementations of LCAP, enterprises will often start with a small project with one team (Kruit, 2018; Richardson et al., 2014; Tiemens & Weel, 2019). In the early stages, a beginning a federated team will have the freedom to design to discover what is possible with LCAP (Richardson et al., 2014). However, when the number of teams increases, the need for consistency and alignment will grow, leading to the need for a central team (Tiemens & Weel, 2019). This central team becomes more dominant and should focus on reusing assets, standardizing, governing, setting norms, and providing architectural practices to promote the desired behaviour (Richardson et al., 2014). Therefore, the need to reduce misalignments between the centre and the federated teams is crucial, as discussed in *subsection 3.3.2*. Across enterprises, the definition of federated development ways of working in IT development teams differs. In some organizations, these models are described as global software development teams, shared service centres, digital factories, communities of practice, virtual teams, or off-shore development teams (Frost et al., 2002; Furst et al., 2004; Kumar & Brouwer, 2020; Landy & Conte, 2013; Marciniak, 2012; Tufekci et al., 2010). However, in practice, when a central entity organizes these teams, similar to in a federated model, a common term is a Center of Excellence or a Center of Expertise (Frost et al., 2002). Therefore, when looking at a central entity on the level of a technology platform or a specific type of knowledge, the definition of the centre of excellence (CoE) is similar to the federated model concept. It can be defined as *'an organizational unit that embodies a set of capabilities that has been explicitly recognized by the firm as an important source of value creation, with the intention that these capabilities be leveraged by and/or disseminated to other parts of the firm'* (Frost et al., 2002, p. 997).

Drawing upon the research on the platform owners and the definition described above, it would make sense that the central team (or the centre of excellence) is also the platform owner of the underlying technology. Moreover, it would make sense that this central team is placed in a position that covers all departments to leverage its capabilities across the entire enterprise. The federated teams could be established in other parts of the firm and, therefore, also be located elsewhere (geographically distributed). In this situation, a federated model for centres of excellence would make sense since it can be used in the best way when an organization is large and globally distributed where the decision making is decentralized (Biggins, 2018). Furthermore, when setting up a centre of excellence, enterprises should decide on the number of business units it wants to cover if it should be country or region-based and if it intends to cover only back-office functions such as HR, Finance, and procurement or also cover front-office functions such as sales (Anagnoste, 2013). Among the more giant multinationals, there are rising more and more centres of excellence while also having problems with the managerial difficulties accompanied by this (Frost et al., 2002).

### 3.3.4 Governing software developments as a central entity in a federated model

To answer **SRQ2**, we need to know how federated software developments can be governed. However, as explained, this can be viewed on various levels. More specifically, there could be federated governance on enterprise-level, functional, and even platform and application levels. In *subsection 3.3.2*, (federated) IT governance in general, is researched. According to Weill & Ross (2004a), governance should be designed at multiple organisational levels when several subunits and functions may be geographically dispersed across an enterprise. More specifically, previous IT principles are defined on a higher level.

When looking at *Table 9*, we see that various governance archetypes can be used depending on the IT decision. In the context of this research, we see the platform owners that introduce LCAP in enterprises, define the IT principles, and allocate budget to the LCAP program as a duopoly. IT specialists in an IT monarchy decide the platform architecture and the IT infrastructure. When looking at the IT decision on business application needs, some top governance performing companies use the federated archetype (Weill & Ross, 2004a). These decisions are all on the platform level. Therefore, when looking at the application level where federated teams are located in autonomous subunits in a decentralized organization, decisions on IT investments, principles and even demand could be decided by another

governance archetype. For example, business needs could be managed on a regional level by local managers.

The research of Andriole (2012) suggests that if enterprises want to be agile while cutting costs, IT managers should change the organization with a centralized infrastructure with decentralized business application developments. As described in *subsection 3.3.3*, in the context of this study, the federated structure's central point is the platform owner of LCAP. Therefore, the federated teams and their members can be seen as users of a service hosted by a central team of the same federated structure. This structure is in line with the research of Anriole (2012). However, when looking at the 'business application need' IT decision by Weill & Ross (2004a), we have to define what tasks are applicable in the context of software developments.

As explained, IT governance is about ensuring that the desired behaviour is pursued (Weill & Ross, 2004a). This can be defined by determining what responsibilities each entity has in a federated model. When looking at the central entity in a federated model, responsibilities and tasks should be defined. In federated governance, there should be coordination of federated teams by providing policies, processes, and informal people-focussed techniques (Williams & Karahanna, 2013). When looking at governing software development, the underlying goal of reaching this desired behaviour in a federated model is to ensure quality within the federated teams (Dahm et al., 2019; De Vries, 2019; Tiemens et al., 2019; van Brummelen & Slenders, 2019). However, seeking quality while having autonomy in starting federated teams also raises a dilemma; on the one hand, formalization will increase a federated team's decision-making and effectiveness (Bourgault et al., 2008). On the other hand, teams should be encouraged to work as autonomously as possible to be successful (Bourgault et al., 2008). Therefore, there is a balance between autonomy and governance or control (Brown, 1999); independence enables federated teams to have design freedom and increases development speed, while governance enables federated teams to adhere to the defined desired behaviour, including processes and standards. Therefore, we have to determine what this desired behaviour is and what tasks and activities should be executed in the context of federated software developments while taking into account this dilemma.

In this research, the platform owner decides on the IT principles, and therefore, we will approach the governance from a top-down approach. Consequently, it is looked at what tasks and activities a central team has in a federated software development context. The next part defines the tasks of a central entity in a federated structure in the context of software developments. A set of papers is analyzed, and the tasks, which are applicable in the context of software development, are extracted and grouped to the tasks that a central team should execute in a federated model.

### *1. Define operating model/ responsibilities*

Since IT governance can be viewed as providing IT rights and responsibilities to enforce the desired behaviour (Weill & Ross, 2004a), these should be defined. As discussed, a federated model is concerned with a model where a central entity is linked to a group of autonomous federated sub-entities. Therefore, we should identify what responsibilities a central and a federated team have in this context. This also should indicate what kind of access both parties have (Seiner, 2017). Furthermore, according to Anagnoste (2013), an operating model should be selected before starting the central team's program.

### *2. Maintain and manage reusable assets*

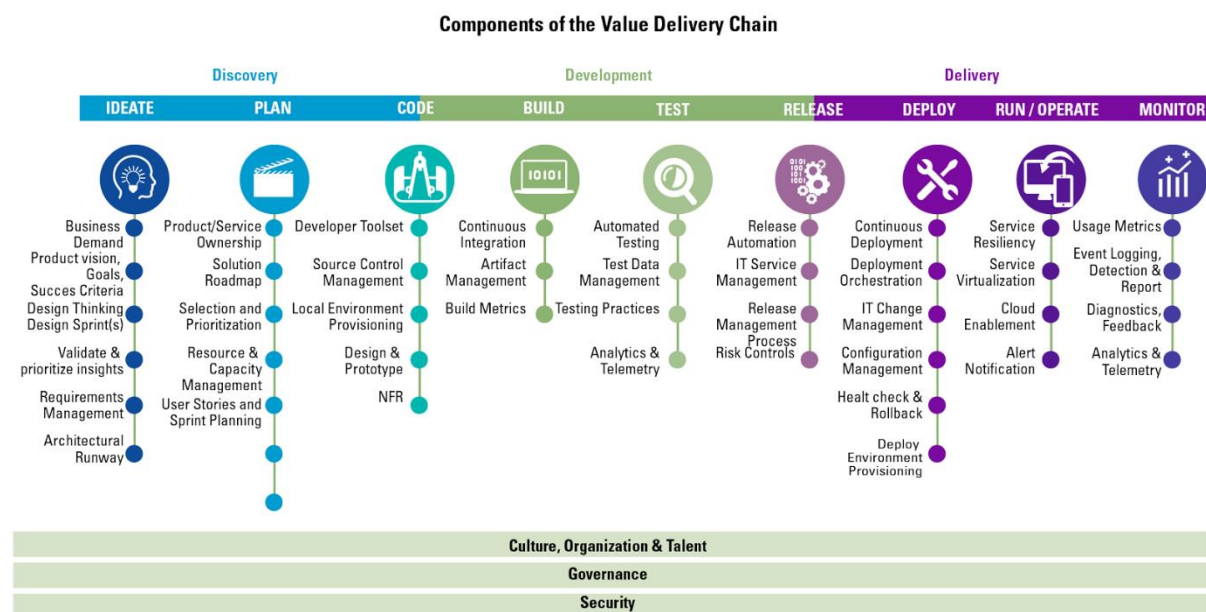
As discussed in *section 3.2*, reusability is a common topic in software development. In parallel, reusable repositories are a common theme in federated models. As a central entity within a federated architecture, it is evident that they should be in charge to manage the repository for artefacts that can be reused (Altintas et al., 2007; R. Fischer et al., 2007). To make optimal use of the assets developed by each federated team, teams should be aligned. The reusable assets could be shown in a catalogue to show what can be created, inspire other subunits, and foster reusability (Afsarmanesh & Camarinha-Matos, 1997; Le Clair, 2017).



Furthermore, when looking at LCAP (as noted in *section 3.2*), software applications that are created in LCAP are based on the same programming language. Hence, it allows users for central repository management and monitoring within the same platform. This means that resources can be shared, and alignment between the federated teams can be reinforced to lower IT costs (Williams & Karahanna, 2013).

### 3. Set guidelines and organizational policy

To ensure that the federated teams follow the desired behaviour, guidelines should be defined, including the process that should be followed, identifying and prioritising new developments and the agreed development standards (Anagnoste, 2013). These processes also include how developers should build the applications and what steps they have to follow. Moreover, policies such as what should be done before deploying the artefact into production should be defined (Anagnoste, 2013). There are various ways to set up guidelines and policies for central entities. These set-ups of operating models can be divided into three dimensions: (i) maturity, (ii) scope, and (iii) delivery model (Anagnoste, 2013). This maturity dimension can be seen as the development lifecycle and can be divided into two subdimensions; build and run (Anagnoste, 2013). Looking at governance, making clear what the desired behaviour is for developing applications and providing transparency is crucial for effective governance (*Table 10*) (Weill & Ross, 2004a). Furthermore, as explained in the introduction of this subsection, formalization does increase team effectiveness, especially when the distribution of teams grow (Bourgault et al., 2008). However, setting too strict formal structures in a centralized way could negatively influence knowledge sharing among federated teams (Tsai, 2002). Therefore, the formal structures should not be too detailed.



**Figure 11:** Typical elements of the lifecycle of modern software development (van Brummelen & Slenders, 2019)

### 4. Enable and manage technologies

In a federated model, the cost can be saved by centralizing software platforms used by distributed teams (S. J. Andriole, 2014). Therefore, the central team should manage the licenses with the software vendors (Le Clair, 2017). Moreover, since the central team is also the platform provider, it should ensure that the software works correctly and that the federated teams can adequately use it (R. Fischer et al., 2007).

Furthermore, during a project's lifecycle, some mechanisms should be in place to minimize risks (Anagnoste, 2013) and increase quality and speed (van Brummelen & Slenders, 2019). An example of an agile delivery process in software development with a set of mechanisms is shown in *Figure 11*. Similar to

the built and run dimensions of the research of Anagnoste (2013), they are also shown in *Figure 11*; the green development stages are part of the building phase, whereas the purple stages are taking care of deployed artefacts. A central entity in a federated model could be responsible for ensuring that these mechanisms are in place. When having mechanisms in place that automate the development process, speed, and quality can be delivered (van Brummelen & Slenders, 2019). This is especially important when teams are testing and deploying frequently (van Brummelen & Slenders, 2019). This automation could also be achieved by introducing testing capabilities on CI/CD pipelines (see *Easy management and monitoring*).

### 5. *Communication and alignment*

As discussed, sharing components and knowing about these components is about aligning federated teams. It is crucial to ensure that federated entities are aligned to reap a federated governance model's benefits (Williams & Karahanna, 2013), as explained in *subsection 3.3.1*. Therefore, the central entity should ensure that the federated teams are aligned and up to speed (Richardson et al., 2014). Without proper communication, a central entity could not work out efficiently (Marciniak, 2012). In addition, as can be seen in *Table 10*, support and communicating is crucial for effective governance.

A high-level strategy, including a goal and purpose distributed to each federated team in the federation, is recommended to align software teams (Dahm et al., 2019). The federated teams joining the federated structure need to share a joint mission and have similar purposes (Lindström et al., 2017). However, this strategy should already be established at the beginning of the software platform implementation. One way to do this is to create an open culture since it stimulates teamwork and openness to learn from each other (Marciniak, 2012; Wenger, 1998). Therefore, teams should be flexible and open-minded. As shown in *Figure 11*, culture is part of modern software development's full lifecycle and should stimulate collaboration and continuous improvement with an entrepreneurial mindset (van Brummelen & Slenders, 2019). Therefore, similar to the characteristics of LCAP (see *section 3.2*), teams in a federated model should be agile and multidisciplinary; when these teams join the federation, they should have some time to adjust to the internal culture and technology (Dahm et al., 2019).

Furthermore, communication with internal people outside of the federated teams could be of high value. Therefore, a change management program could be set up to promote technology (Anagnoste, 2013). Showing the platform capabilities and developed products can trigger other parts of the enterprise to start developing in the same technology or reuse created software (Marciniak, 2012). Other examples to bring teams together and to align are hackathons. Hackathons can be held to bring developers together to learn from each other while co-developing artefacts (Dahm et al., 2019).

### 6. *Knowledge management and educate*

The teams' alignment is also vital to ensure that information is shared among the other federated teams. This is the management of knowledge to make sure federated teams work efficiently. The central team should foster the federated teams' work to increase the organisation's progress as a whole (Marciniak, 2012). According to Marciniak (2012), knowledge management is a significant characteristic of centres of excellence. Knowledge management also includes training to increase the knowledge of the federated team members. Therefore, training on best practices could be provided by the central team (G. Fischer et al., 2004; Richardson et al., 2014). According to Fischer et al. (2004), training on best practices by experts to beginners is one of the most critical aspects to ensure success. Likewise, training, technical and management support are also extremely important for success (G. Fischer et al., 2004). Giving training could increase the quality of the end-deliverables in federated teams. By focusing on quality in software developments, teams are more likely to end up with less technical debt (Dahm et al., 2019).

Knowledge management also covers sharing best practices among federated teams, which is one of the central team's core activities (Dahm et al., 2019; Marciniak, 2012). Moreover, after teams are used to working with LCAP, architectural best practices should be provided to ensure that applications are built on a sound basis (Richardson et al., 2014). Providing best practices is different from reusing components since

best practices could also be, e.g. ways to approach projects, set up teams, architectures, or training schemes.

### 7. *Prioritizing and assessing projects within a company*

According to Anagnoste (2013), one of the key questions when having a central team is to decide how new projects should be identified and prioritized. As discussed, a federated governance model is about aligning federated teams while keeping a level of autonomy (Brown, 1999; Williams & Karahanna, 2013). The more independence these teams have, the more responsibilities and control they have. Therefore, looking at the application level, when IT investments are made in a federated archetype by the federated team, this goes hand in hand with the autonomy of prioritizing software developments. This prioritization is also dependent on the governance archetype when looking at IT investment decision (Weill & Ross, 2004a). Hence, it is dependent on the governance and operating model whether the central team should prioritize or only advise on prioritization and assessment of the projects. Furthermore, according to Prikladnicki & Yamaguti (2004), software developments should be assessed to understand the risks involved. This assessment should also occur after the development to determine whether a project was successful or not (Prikladnicki & Yamaguti, 2004). These results should be distributed among the teams as lessons to learn, even if it was unsuccessful.

### 8. *Manage operations of deployed assets*

According to Le Clair (2017), a central entity should take care of the production environment's operations when developing a federated governance model. In addition, as discussed in '*Set guidelines and organizational policy*' above, the development process can be split into two stages; build and run (Anagnoste, 2013). The last stage is when the products are already developed and deployed to the production environment. Here, activities such as making sure that the operations run without any errors occur (Anagnoste, 2013). Moreover, continuously monitoring, developing key performance indicators (KPI's), and performing security and compliance checks are the subject of this stage (Anagnoste, 2013). Therefore, according to Anagnoste (2013), clear segregation of development, deployment and monitoring should be established. As shown in *Figure 11*, this separation is also shown in the entire software development lifecycle by van Brummelen & Slenders (2019).

According to Anagnoste (2013), in a Robotics & Process Automation (RPA) CoE, developers building the robots on the platform in enterprises should never be able to have access to the production environment. To ensure that the developed robots comply with the correct rules and standards, a control mechanism could be put in place in the development process before it goes to the production environment. One way to do this is to do user acceptance tests (UAT) that the end-user should sign off and involve all crucial stakeholders in the process before the go-live moment (Anagnoste, 2013). This can also be based on a control framework, which should be set up by the central team too (Le Clair, 2017). However, this exclusion for developers in the production environment contradicts the low-code development DevOps characteristic (as explained in *section 3.2*), where developers also take care of the operations. Therefore, this responsibility should be further investigated.

To conclude, to govern federated software development teams in the context of LCAP, central teams are often also the platform owners that set the guidelines. Therefore it makes sense to look at the responsibilities of a central entity in software development. Hence, a comparison was made with centres of excellence focusing on bringing together and coordinating capabilities in one central place. The list sums up all activities and responsibilities of a central team in a federated software development governance model and is shown in *Table 11*. From a governance perspective, these responsibilities and tasks should be taken into account by a central team when governing a federated software development operating model successfully.

**Table 11:** Responsibilities of the central team in a federated software development model

Nr.	Responsibilities and actions	Source
1.	Knowledge management and educate	(Anagnoste, 2013; Dahm et al., 2019; G. Fischer et al., 2004; Le Clair, 2017; Marciniak, 2012; Richardson et al., 2014; Weill & Ross, 2004a)
2.	Define operating model/ responsibilities	(Anagnoste, 2013; Seiner, 2017; Weill & Ross, 2004a)
3.	Maintain and manage reusable assets	(Altintas et al., 2007; R. Fischer et al., 2007; Kirschner & Roth, 2014; Le Clair, 2017; Marciniak, 2012; Seiner, 2017)
4.	Set guidelines and organizational policy	(Richardson et al., 2014; Seiner, 2017; van Brummelen & Slenders, 2019; Weill & Ross, 2004a)
5.	Enable and manage technologies	(R. Fischer et al., 2007; Le Clair, 2017; Seiner, 2017)
6.	Communication and alignment	(Anagnoste, 2013; Seiner, 2017)
7.	Prioritizing and assessing projects within a company	(Anagnoste, 2013)
8.	Manage operations of deployed assets	(Anagnoste, 2013; Le Clair, 2017)



## 4. Explorative case study

As discussed in *chapter 3*, it was found how organizations can govern software developments in a federated way and what low-code characterizes. In this chapter, the following research questions will be answered:

- **SRQ3:** *How does the low-code paradigm relate to federated development?*
- **SRQ4:** *What mechanisms, tasks and best practices can be used to govern federated development teams under a low-code paradigm?*

First, the interview setup including the details of the interviewees will be presented in *section 4.1.1*. Thereafter, **SRQ3** will be discussed in *section 4.2*. Lastly, mechanisms, tasks and best practices were identified that improve federated low-code governance during the interviews with consultants, low-code vendors and other companies. This will be discussed in *section 4.3*.

### 4.1 Interview setup

In *subsection 2.2* under *Interviews*, the research method was introduced. In this section, it is discussed in more detail how this research was conducted. To holistically look at how federated low-code models should be implemented, the interviews were conducted with a broad set of interviewees consisting of IT consultants, low-code platform vendors, and enterprises that already implemented a similar federated software development model. Identical to *chapter 3*, for the interviews with the low-code vendors, only 'leading' LCAP and one platform labelled as 'visionary' by Gartner (Vincent et al., 2020) were selected. In total, 14 hours, 20 minutes and 35 seconds of interviews were conducted with an average of 1 hour, 1 minute and 28 seconds per interview. Since a federated approach to low-code teams was not common, interviewees of companies with federated operating models that use similar software development platforms were interviewed. In this way, different cases across various industries were analyzed to explore their operating models, mechanisms, best practices, processes and tasks.

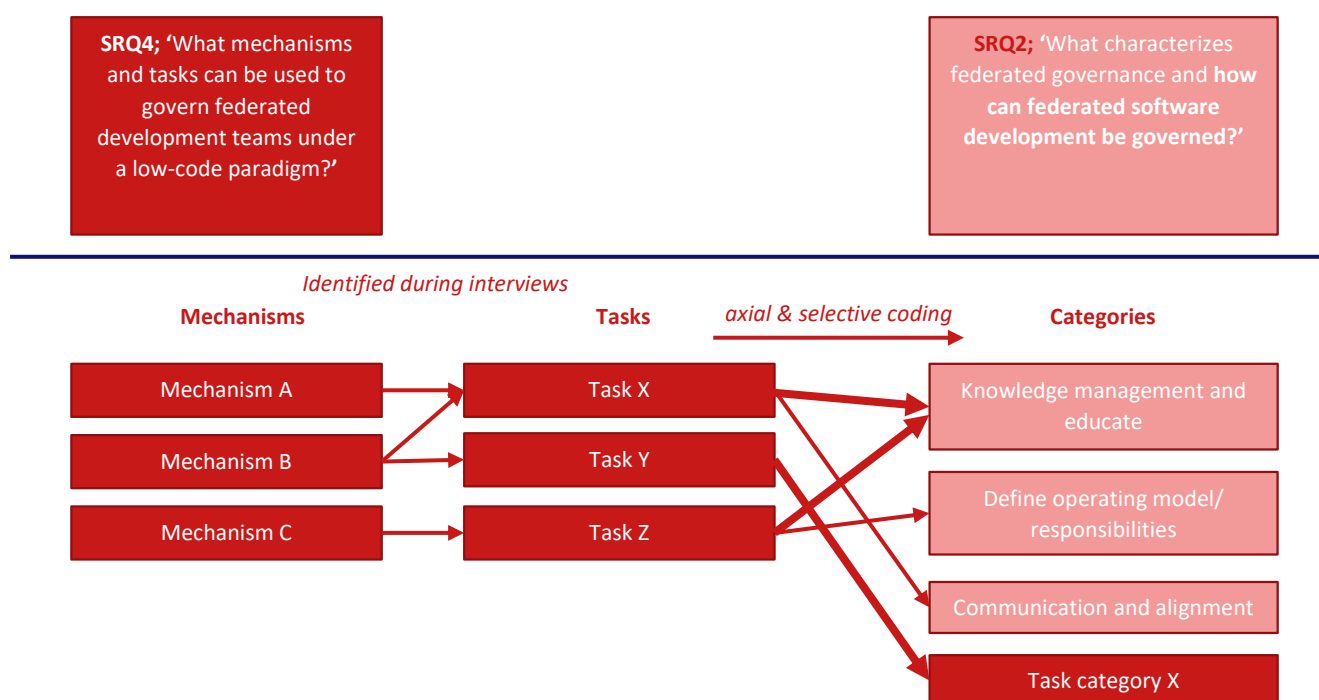
#### 4.1.1 Semi-structured interviews

As shown in *Table 12*, a broad set of enterprises and consultants are interviewed to receive an objective view. Both these type of interviews will substantiate and add value to answering the sub-research questions.

**Table 12:** Interview list

Nr.	Resp.	Industry	Role description
1.	Respondent K	Low-code consultancy	Manager Solutions and Technology
2.	Respondent M	Low-code consultancy	Chief Technology Officer
3.	Respondent C	Technology consultancy	Head of Intelligent Automation CoE
4.	Respondent B	Technology consultancy	Partner - Operating models - Intelligent Automation
5.	Respondent A	Technology consultancy	BeNeLux Leader of Automation Services
6.	Respondent G	Low-code vendor	Global Senior Director, Digital Transformation
7.	Respondent H	Low-code vendor	Chief Technology Officer
8.	Respondent I	Low-code vendor	Chief Executive Officer
9.	Respondent N	Low-code vendor	Senior Solution Architect
10.	Respondent J	Banking	Head of RPA & Intelligent Automation CoE
11.	Respondent F	Logistics & chemicals	Architect CoE low code
12.	Respondent D	Oil & Gas	DIY Implementation Lead
13.	Respondent L	Banking	Head of Center of Excellence Robotics Process Automation

As explained in *section 2.2*, a template approach is used as input for the coding process for answering **SRQ4**. This approach is schematically shown in *Figure 12*. The elements in *Figure 12* present how the coding process was executed on an abstract level. Yet, more categories were identified. The mechanisms and tasks that were identified are illuminated in dark red. As can be seen, the output of the literature review of **SRQ2** (*Table 11*) is used and shown in the light red boxes. During the explorative case study, the transcripts were coded and these light boxes served as a starting point. Nevertheless, this did not limit the coding process in adding new categories. Therefore, as can be seen in *Figure 12*, new categories could be added (see ‘Task category X’), or existing ones could be changed. Furthermore, during the coding process mechanisms were also identified that support the tasks that should be executed by the central team. Moreover, as shown in *Figure 12*, some mechanisms will support some tasks, and some tasks will fall under multiple categories. However, only one task was dominant which is indicated with a more robust line. Therefore, the identified categories are not mutually exclusive.



**Figure 12:** Set up of SRQ3

## 4.2 Relation low-code paradigm to federated governance

In this section, sub-research question 3 (**SRQ3**), ‘How does the low-code paradigm relate to federated development?’ is answered. For the consultants, only interviewees were selected that have experience with low-code, which were respondent B, G, H, I, K, M, N. By interviewing low-code vendors and low-code consultants, it was analysed how the low-code paradigm and its characteristics listed in *Table 8* related to a federated governance model. During the interviews, no characteristics other than the already identified characteristics in the section ‘The low-code paradigm’ were found. Most of the characteristics listed in *Table 8* and the limitations of low-code development were supported during the interviews. There were no characteristics unsupported. The identified relations were open coded, and later axial coded. The coding schemes can be found in *Appendix IX* and the found relations are presented in *Table 13* and briefly discussed below.

**Table 13:** Low-code development relates to a federated governance model

Nr. Topic	
<b>Benefits and opportunities</b>	
1.	Transparency in applications (4 respondents)
2.	More local developers (4 respondents)
2.1	New developers in all segments & cross-functional teams (2 respondents)
2.2	No full-stack developer or long training needed (4 respondents)
3.	Monitoring and access capabilities (4 respondents)
4.	Quality assurance (2 respondents)
5.	Quick local development (7 respondents)
<b>Risks and limitations</b>	
6.	Compliance and security (3 respondents)
7.	Lack of adaptation of a new way of working (5 respondents)
8.	Lack of expertise (5 respondents)

#### 4.2.1 Benefits and opportunities for a federated low-code model

First, a set of found benefits concerning federated low-code development is provided per category.

##### A. Transparency in applications (4 respondents)

As described in *Table 8*, LCAP make use of easy to use drag and drop interfaces to build and review applications from various layers (deployment layer, data layer, service integration layer, application layer). Moreover, the platform allows developers to use deployment features and collaboration tools within one platform (see *Easy management and monitoring* and *Collaboration*). Because of these characteristics, central teams can support federated teams easier because it is easier to do a deep dive into applications to review. Therefore, when studying or evaluating projects, developers do not have to talk about the actual coding underneath, which creates an easier understanding of the topics to review. In addition, when having multiple federated teams, peer-reviews among federated teams that increase quality can be conducted easier. Furthermore, as explained in *section 3.2*, since the DevOps way of working is supported within LCAP, incremental and iterative development (IID) is possible. Therefore, short code reviews can be done that allows a central team to give insights into federated teams' progress, which is in line with the research of Paasivaara & Lassenius (2004).

*'So it's also making it more accessible for centres of excellence regardless of their role, right. So whether you're a UX designer or an architect, you should be able to understand what the low-code platform is doing. And that's going to also be helpful in these Federated models'* - Respondent N, Senior solution architect, low-code vendor

##### B. More local developers (4 respondents)

As explained in *section 3.2*, developing with LCAP allow enterprises for attracting a new type of developer. Therefore, enterprises will have a bigger talent pool of software developers. In this way, it becomes easier to start a federated team because it is relatively easier to set up a team with people that can build software applications. Consequently, it lowers the barrier to set up teams because it becomes cheaper since federated teams do not have to be solely filled with expensive full-stack software developers.

Furthermore, because more teams can be set up, more applications can be created and reused among other federated teams and more knowledge and best practices can be shared when the federation is growing. This enables organizations to deploy software developments quicker and to enhance digital transformation.

Moreover, since different types of users can come from all kinds of business domains, cross-functional teams can be set up. These new users can come from business. Therefore, the focus shifts from technical aspects in software development to more customer-orientated applications that focus on business rules and business processes. Because of the federated governance, more local knowledge can be used to develop customer-oriented applications quickly.

*'But low-code needs that federated model because then you can actually do where the value of low code starts to shine; if you have small autonomous teams, cross-functional teams, that are focused on the domain knowledge, then you can live up to the promise of this additional speed; you can get up to 10 times faster and as well as that more people can be involved in software development instead of only computer scientists'* – Respondent H, Chief Technology Officer low-code vendor

### **C. Monitoring and access capabilities (4 respondents)**

As explained in *section 3.2*, in LCAP it is easy to manage roles and authorization within the LCAP itself. Therefore, developers can be assigned to federated teams and can be restricted from executing specific actions depending on the type of developer role. Therefore, a central team can govern these types of developers easily in one platform. Furthermore, most LCAP offer the possibility to monitor all applications and performance in one central overview. Moreover, since most leading LCAP are also offering their platforms in the cloud, projects can be viewed online and reviewed in real-time within the platform by the central team or other federated teams. Because of this federated structure, central teams can set up the cloud infrastructure to improve and monitor performance centrally.

*'There are also associated management tools with that, so you are more able to monitor from a distance than you are from the perspective of a traditional programming environment. So there are more tools there for a centre of excellence'* - Respondent N, Senior solution architect low-code vendor

### **D. Quality assurance (2 respondents)**

Because of the pre-built coding blocks and code generation, where the platform codes without a developer, a level of quality are already built-in. In this way, federated teams will automatically develop applications of a specific quality level. In addition, the platforms also allow configuring these building blocks in such a way that particular users only have access to a specified set of components. Furthermore, as identified in *section 3.2*, LCAP consist of all kinds of capabilities within the platform, e.g. a code review tool that reviews technical errors (not on the functional design). Therefore, when the code is not sufficient, the code cannot be deployed to the production environment. Moreover, when making it mandatory to implement security modules or components such as single sign-on<sup>7</sup> (SSO) make it easier to build in a layer of security. Therefore, central teams can have some trust in these federated teams since specific standards can be implemented relatively easy. However, it should be noted that the quality of these pre-built building blocks does not imply that the functional design of an application is of high quality, but merely on the technical side.

*'Quality assurance, that's already in the platform, and it starts with that this should already give a Center of Excellence, or whatever that team is called, faith in the solution. There's sort of a minimum level that is going to definitely be reached'* - Respondent N, Senior solution architect low-code vendor

*'That is the beauty of low-code, right? You don't code, but the platform codes. If the platform codes, then inherent, there is a lot of security already in the platform'* - Partner Intelligent Automation, Technology consultancy

### **E. Quick local development (7 respondents)**

As elaborated in *section 3.2*, the low-code paradigm is characterized by a new development method such as DevOps development. More specifically, in the DevOps methodology, federated teams are also responsible for the operations part when the applications are deployed to the production environment. During the interviews, the advised way of working to develop in a federated low-code model is examined, when looking at the SDLC operations part. As shown in *Table 14*, all respondents but one suggested letting federated teams take care of the operations themselves. Therefore, DevOps teams are recommended in federated low-code model and will be used in the design of the artefact. The reasoning why this way of working will be used is explained in detail below.

---

<sup>7</sup> Single Sign-On is a commonly used component in enterprises that allows users to login into applications via once via one single account that gives access to a set of applications.



**Table 14:** Interviewees on using DevOps in federated low-code teams

Respondent	B	G	H	I	K	M	N
Answer	✓	✓	✓	✓	✓	~	✓

Similar to the research of Anagnoste (2013) and Le Clair (2017) as discussed in *subsection 3.3.4*, Respondent M, a low-code consultant, mentioned that a central operations team was also an option for a federated low-code operating model next to the option of DevOps. In this way, operations of developed applications are carried out centrally or via an external partner (Anagnoste, 2013). In these operating models, applications will be handed over to a central operations team after the deployment. In this way, there is a responsibility transfer to a central team that should ensure the application will run. When there are bug fixes, the central team would be responsible to resolve this and support all developed applications. Proponents of this operating model say that this model is superior since operations are monitored 24/7 by operation experts that only focuses on the applications that are in the production environment. Furthermore, developers can only focus on building their applications; when taking away the time spent on operations, it can be spent developing. Additionally, in teams with a small number of developers that are also focusing on the operations of their applications, it could have large consequences when this developer leaves the company. In that case, there is a risk that there is no developer that can support the application. Likewise, when having a central operations team, federated teams should do a hand-over to the operations team to check if the quality is sufficient to take the application over to their team. However, as can be seen below, this also creates issues.

*‘This is an issue in IT where you have to developers building something and they just sort of throw it over the wall to the operations, and then the operations has issues. They will say it works in my environment and you have a lot of communication issues there’* – Respondent N, Senior solution Architect, low-code vendor

Furthermore, if operations are carried out centrally, this handover will increase the barrier to develop applications since there is an extensive quality check in the pre-deployment stage before the operations team allows the application to be brought in. This does not foster the entrepreneurial mindset that low-code countries try to offer by rapid development and a short time to market.

Nevertheless, these models where the operations are handled centrally, as proposed by Anagnoste (2013) and Le Clair (2017), were designed for another software technology; robotics and process automation (RPA). In RPA, processes are automated, which, in general, does not need frequent updates or features added. However, in a low-code development context, applications are built where features have to be added regularly and where repeated deployments can occur. Therefore operations and development are close together in these platforms.

*‘Yeah, but I think there’s not such a big difference between the development and the operations in a low-code platform. In the low code platform, that’s very close together. It’s all about the business logic that you build in your app’* – Respondent H, Chief Technology Officer, low-code vendor

As can be seen, LCAP make it easy to use a DevOps and agile way of working by automating the deployment process as much as possible. This is also admitted by respondent M, who previously also mentioned central operations as a possibility;

*‘The platform takes care of a lot for you of that stuff, so you don’t need to know all those nitty-gritty console commands and scripts to be able to push something into production. It is really as easy as next, next, finish<sup>8</sup>. And that’s like the major advantage’* – Respondent M, Chief Technology Officer, low-code consultancy

<sup>8</sup> ‘Next, next, finish’ is slang in software engineering for applications such as an installer where you only have to click on next, next to install a package. This indicates the ease of use of phenomena within LCAP.

Therefore, these LCAP are designed to use DevOps teams that can use these platforms. Consequently, almost all respondents recommended DevOps teams that are fully responsible for their developed applications in a federated development context. They mentioned various reasons why this model is advised. First, according to respondent K, using a central operating model would not increase the federated teams' quality because they will not learn from their own mistakes. For example, when errors are arising, and the teams have to solve them themselves, they will learn from them.

*'And, of course, you can say well, I'm only going to develop these applications and my Mendix partners are going to support it, so I don't need DevOps. Well, that's true, but if you want to be able to guarantee the quality of your applications in general, you should probably do DevOps'* - Respondent K, Manager, low-code consultancy

Hence, compared to traditional software development, in LCAP, it is easy to deploy, update, and even do infrastructural management, unless the LCAP are not running on-premise. Consequently, according to respondent H, there is not much to do in operations.

*'So most of the operations is part of the platform, so that's automated. There's not a lot you have to do if you have to deploy it, and then basically it runs, and we [the platform] are operating the platform'* - Respondent H, Chief Technology Officer, low-code vendor

As stated in *Table 8*, partly thanks to DevOps, low-code development is characterized by speed because projects can be started and realized reasonably quick. Similarly, when having autonomous federated teams, they are independent in a way that they can decide on their project. Consequently, decisions on projects are taken on a more local level. Hence, some central political business processes can be bypassed, and decisions can be made rapidly. In this way, the barrier is lowered to start building applications and be able to respond to business demand faster by shortening the time to market and being flexible. Likewise, because people from the business could also be part of the federated team, there is a close leap between business and development. Therefore, a federated governance approach will support low-code DevOps teams.

*'The operations are part of the federated team; that's why we talk about DevOps. So having not only built that application but also run that application. So, you build it, you run it, is the vision there. And even on top of that, because the Federated team is so close to the business, you have a very small leap to talk about business DevOps'* – Respondent G, Global Senior Director, low-code vendor

#### 4.2.2 Risks for a federated low-code model

During the interviews, the limitations of low-code in a federated governance model were also discussed. In a federated model, teams are autonomous and should operate without many dependencies on the central team. Therefore, due to this autonomy level, there are risks accompanied in a federated low-code model, which are grouped into four categories and described below.

##### *F. Compliance and security (3 respondents)*

Control on security on the application level is more complicated to do from a central point of view in a federated model. For example, it is difficult to have a picture of what type of data is processed in each application. It is always hard to check on this, especially in a federated model, because there is less control from the central team, and the teams can consist of new developers that do not know about the risks of development. Therefore, in centralized software development, there are fewer problems to risk management (Prikladnicki & Yamaguti, 2004). Thus, control mechanisms have to be implemented to ensure a certain level of security. Components such as SSO will already improve the security level (see *footnote 7*). This risk is in line with the findings of *section 3.2*. However, to be compliant, there needs to be a check to ensure the federated teams do not use sensitive personal data or data that is not compliant. Therefore, there is a need for monitoring these federated teams and the applications that are being developed.

*'So when people are building Mendix applications, and especially not that experienced people, you notice that security is one of the biggest problems that there is in an application'* – Respondent K, Manager, low-code consultancy

### G. Lack of adaptation of a new way of working (5 respondents)

When having new platforms, a new way of working and a culture is needed (Richardson et al., 2014). Similarly, when implementing a low-code federated model, companies have to adapt to new cultures and mindsets. As described in number E, the DevOps method is the recommended way of working with federated low-code teams. In addition, as explained in *section 3.2*, agile development teams are also usual in low-code development. Consequently, because the central team manages all platform and infrastructural issues, there is a more significant risk that teams do not understand the new way of working or not using the full potential of DevOps or agile methods for low-code.

*'You can't really do low code in a traditional way, so you can't do waterfall in low-code. Like; everything is possible, but that's a terrible idea. It's going to lead to a disaster. A low-code environment sort of requires an agile way of working'* -

Respondent N, Senior solution architect low-code vendor

### H. Lack of expertise (5 respondents)

According to the respondents, some organizations or managers still think that low-code is a tool for simple applications that are more for fun. They are not aware of the platform's capabilities. When implemented in organizations, the full potential is not reached and can be perceived as shadow IT. In addition, because in a federated model, the central team takes care of several aspects of the development (e.g. platform maintenance and infrastructure), there is a low barrier to start a team. However, managers who will start implementing these teams should also be aware of the platform's capabilities to use the full potential. Otherwise, there is a risk that the platform or the applications built on the platform are not used in the right way.

*'I think the biggest issue for Centers of Excellence and federated models is probably that a lot of the time low-code is going to be sort of shadow IT at the beginning for organizations. So it's going to be bought by a specific business unit for a specific purpose, and it's not necessarily going to be part of the IT organizational structure, and I think that that's probably the biggest risk in terms of having low-code within your organization'* - Respondent N, Senior solution architect low-code

vendor

When having a federated governance model, low-code teams can start a team that will be, most of the time, initiated by specific business units that see an opportunity in the technology. However, these federated teams could consist of starting developers that got appointed by the managers. As stated in *subsection 3.2.1*, these new users have less experience in development than full-stack developers.

Similar to the findings of *subsection 3.2.1-5*, the interviewees mentioned risks regarding software developments. From the interviews, it became clear that partly because of the type of users, there should be a higher emphasis on explaining the development process by setting up a guardrail. These guardrails should be the development guideline for developers in federated teams and are in line with the central responsibility as defined in *'Set guidelines and organizational policy'*. This lack of expertise could lead to worse designs.

*'I think the limitation is that it only works with the rights level of people, or enough people that are at a certain level, so to be independent and the right culture.'* – Respondent H, Chief Technology Officer low-code vendor

To conclude, governing low-code development teams in a federated way has some opportunities such as the DevOps way of working, tools and monitoring capabilities built in one platform, and having a bigger talent pool throughout the enterprise that can also focus on developing local applications. In contrast, there are also risks involved in governing low-code development teams in a federated way. The main risks are that new users can build applications and, therefore, there are higher risks in having security and compliance issues. Furthermore, enterprises should understand the new DevOps, agile and unique type of development. Since DevOps is the recommended way of working in a federated low-code model, the operations task, as identified in *subsection 3.3.4* (the last item in *Table 11*), is assumed to not be relevant in the context of a federated low-code governance model. To understand how to mitigate the risks

identified in the interviews and the literature study, interviews with respondents across industries with similar federated software development models were conducted as well as with technical consultants. During these interviews, processes, tasks, best practices, and mechanisms were identified and will be discussed in *section 4.3*.

### 4.3 Mechanisms, tasks and best practices of federated low-code development teams

As shown in *Table 11*, a set of universal tasks for a central team in software development, in general, was found. This brings us to a point where we have analyzed federated structures, defined the low-code paradigm and analyzed its relation to a federated governance model. In this section, it is researched if these tasks also apply for a central team, specifically in a federated low-code model and what mechanisms could be used to support these tasks in a federated low-code model. In addition, using these interviews, it is researched how these federated low-code governance models could be best set up, what aspects in these models were most important and what other factors these decisions influenced. Also, other interviewees from industries were interviewed to discover how these processes, structures and mechanisms look like in practice. The questions were oriented to find tasks, processes and structures for a central team that would support implementing a federated low-code model. From this information, a governance framework could be designed. However, before creating a governance model for federated low-code teams, a strict definition had to be defined.

#### 4.3.1 Definition of federated low-code teams

As explained in *chapter 3*, according to Roth (2014) and Sheth & Larson (1990), federated entities have three main characteristics; distributed, heterogenous and (semi)-autonomous. In the context of this study, low-code will be added. These characteristics are shortly described in *Figure 13*, followed by a definition.

Distributed	Heterogenous	(Semi-)autonomous	Low-code
<ul style="list-style-type: none"> <li>Geographically distributed teams</li> <li>Distributed functions</li> <li>Distributed per region</li> </ul>	<ul style="list-style-type: none"> <li>Organizational level: different team sizes and focus domains</li> <li>Team level: different levels of experiences within team</li> <li>Application level: different types of applications within and per team</li> </ul>	<ul style="list-style-type: none"> <li>Teams can autonomously decide on functional design, budget, and planning</li> <li>The teams can autonomously decide on the prioritization of developments and hiring new people</li> </ul>	<ul style="list-style-type: none"> <li>The teams use leading low-code platforms to develop applications</li> </ul>

**Figure 13:** Characteristics of federated low-code teams in the context of the study

Thus, this research aims to find governance structures for starting distributed, heterogenous, autonomous low-code teams. Therefore a clear definition for federated teams in the context of this research is provided in *Definition 2*.

*Starting autonomous, distributed DevOps teams consisting of developers from various disciplines with different software developing experience that are creating solutions for enterprises using a low-code application platform*

**Definition 2:** Federated teams in the context of this study

#### 4.3.2 Federated low-code governance dependencies

During the first interviews, it became clear that the control mechanisms, tasks, and governance operating models depend on a set of factors. Some elements were dependent on the strategy and type of company, whereas other factors were dependent on internal factors such as team maturity and the kind of application. Interestingly, similar to what was found in the literature on the balance between autonomy and control by Brown (1999), interviewees stressed that the federated teams should develop as freely as possible with a central team's level of control. Therefore, an important task is to focus on this autonomy part of federated teams and support them where needed. The coding scheme of this research can be found in *Appendix IX*.

*'If you are not very strict on or keeping your teams autonomous and as independent as possible, it becomes really hard to manage it, and especially if you look at larger companies that want to build software in a successful way'- Respondent H, Chief Technology Officer, low-code vendor*

*'You need to find a balance between centralized governance and autonomy'- Respondent M, Chief Technology Officer, low-code consultancy*

*'If the central team does their job right, they make sure that the teams can run without the central team. The job of the central team is to make sure that they're not necessary' – Respondent N, Senior Solution Architect, low-code vendor*

*'I think, in principle, the centre of excellence is overhead that you want to minimize. It should maybe end up as the third line of defence quality standard' – Respondent B, Partner, Intelligent Automation consultancy*

The interviews came down to four main categories that influence the amount of governance control a central team should have over the federated teams. These identified dependencies are categorized and shown in *Figure 14*. Each category will be briefly described.

Type of technology	Type organization	Type of application	Maturity of team
<ul style="list-style-type: none"> <li>• Knowledge needed</li> <li>• Development method</li> <li>• Type of users</li> </ul>	<ul style="list-style-type: none"> <li>• Culture</li> <li>• Focus on compliance</li> <li>• Operating model</li> </ul>	<ul style="list-style-type: none"> <li>• Complexity</li> <li>• Type of data               <ul style="list-style-type: none"> <li>• Read-write</li> <li>• # of integrations</li> </ul> </li> <li>• Roles &amp; logic</li> <li>• Runtime &amp; criticality</li> <li>• Size of application</li> <li>• # of development lifecycles</li> <li>• Internal-external facing</li> </ul>	<ul style="list-style-type: none"> <li>• Team composition</li> <li>• Type of user</li> <li>• Background</li> <li>• Certification</li> <li>• Internal trainings</li> <li>• Type and # of applications developed</li> </ul>

**Figure 14:** Dependencies for central influence in a federated low-code governance model

### Type of technology

The type of technology depends on how teams should be governed. There is a difference between no-code, low-code and RPA, as was elaborated in *subsection 4.2.1* about the research of Anagnoste (2013). For some technologies, more knowledge is required compared to others which also influences the operating model and how easy, for example, it is to centralize or decentralize operations.

*'There is a difference between how much technical knowledge you need to have in a certain tool that can have an impact on the operating model for a Federated Model' – Respondent A, Benelux Leader Automation Services, Technology consultancy*

### Type of organization

Organizations influence the way how development processes are governed. For example, some companies could behave more entrepreneurial to allow their employees to build applications without much control and intervention. Other companies should take compliance risks more into account, such as the banking industry, where data privacy plays a more significant role. Therefore, dependent on the organisation's industry, there could be a higher focus on compliance which leads to more control mechanisms and oversight from a central team to review developed applications. Others might contain employees with high technical knowledge on the platforms where they trust their employees to build applications without many checks. These companies are more entrepreneurial-minded. Therefore, a company's culture would influence the impact that a central team wants to have over the federated teams.

Furthermore, the strategy and characteristics of the company could determine what role the central team should have. More specifically, if a company has a strong influence from the central headquarters, there could be a central portfolio owner that orchestrates the applications developed by all federated teams. In more decentralized organizations, the federated teams are authorized to select, prioritize and maintain their applications without any formal approval from the central entity.

*'But that also depends on the operating model, like the general operating model of the company; if there is a build, change, run operating model versus an agile operating model. Mostly in the Agile operating model, the principle is; "you build it, you own it" – Respondent B, Partner Intelligent Automation, technology consultancy*

### *Type of application*

The type of application that is being developed could also influence a central team's impact over a federated team. More specifically, the more complex an application is, the more control is needed. This can be divided into a set of factors. These factors are, most of the time, influenced by one and another. For example, if an application is internal or external facing, different measures have to be taken. In the case of external-facing applications, there is a higher need for penetration testing. Moreover, it should be noted that when an application is external or internal facing, it does not imply that it is also more complex. However, in all likelihood, this will be the case.

Furthermore, the control is also dependent on the development cycles needed for an application. If an application needs to have frequent updates or features have to be added, there is a higher demand for automatic deployment options without permissions from a central team. Next, the runtime of the application could influence the control of the central team. In other words, when an application has to run 24/7 with support, it is likely that the central team should help with support if the federated team does not have the capabilities to provide continuous support.

When looking at the application's complexity, this can be divided into other elements such as multiple roles and logic, the size of the application, and the type of data and integrations. Moreover, the size could influence how the development should be. For example, the foundation and architecture of an app differ if the users' volume will increase. In other words, if multiple countries are going to use the same application, there could be a higher focus on a multi-tenant architecture where the central team should have a greater share of the control.

Next, the factor that determines the complexity of a project within this category which is mentioned most during the interviews (7 respondents) is the type of data and access to integrations. During the interviews, the respondents stated that the type of data would influence what kind of control a central team should have. It depends on the type of data, the amount of data, the amount of integrations, risks to losing the data, and if the app only reads or also writes data. Some respondents (respondent B and N) even mentioned the segregation of duties<sup>9</sup>; if the application processes data where segregation of duties is required, such as handling pension schemes. Therefore, governance depends also on the business criticality of the application. In addition, when federated teams are going to use a particular type of risky data, a central team should mitigate the risk that the application would impact the organization financially, legally, or performance-wise. Therefore, to understand the complexity, questions such as 'what is the impact of the application?', 'What happens if the application fails?', 'Does it influence our customer satisfaction?', 'What happens if we do not implement the application?', 'What is the importance of the application or how much does it increase our productivity?' should be asked. Interestingly, a zoning model was used as a mechanism in the oil&gas company to categorize projects, which will be later discussed.

*'We do restrict certain things so you cannot share your application with loads of people in the green zone, you need to follow a certain process for that, so it's really blocked. You can only use a certain set of connectors that we've identified that can't really do a lot of damage'* – Respondent D, DIY implementation lead, Oil & Gas company

### *Maturity of the team*

The governance model and the central control also depends on the maturity of the team. This maturity depends on the team composition and the types of users. These type of users can have various levels, ranging from a beginner to an expert developer. However, to determine what the exact level is, multiple factors could play a role. These factors are described below.

*'And the more mature a team is, the less the centre of excellence is needed'* – Respondent B, Partner, Intelligent Automation Consultancy

---

<sup>9</sup> Segregation of duties or separation of duties is a principle in software development where more than one person is needed to carry out a task to guarantee safety of data or protecting a system.

The team composition is one factor that indicates what a team can handle; if a federated team consists of numerous developers, testers, a product owner and a tech lead with some full-time business leads, it could have the capabilities to build good apps. For example, when a federated team contains an architect, there is less need for supporting a federated team in defining the architecture of an application.

*‘So when the knowledge of the team expands, then you will be able to do less control from the centralized team, and they can handle it themselves’ – Respondent K, Manager, low-code consultancy*

In addition, maturity also depends on individual level per role. An expert developer can build well-developed applications and the central team does not need to check, control and guide her or him. Depending on the expert level, fewer rules from the central teams could be required. However, this definition of ‘an expert developer’ could be based on a set of factors. First, during the interviews, it is found that this can be based on the developers’ background.

*‘You can be a really good developer if you’re not trained officially as a software engineer, but it helps a lot. I mean it is a profession’ – Respondent B, Partner, Intelligent Automation Consultancy*

Secondly, the development experience determines the level of a developer. This can include the number of years of experience that a developer has (3 respondents). In other words, when a developer builds applications for more than ten years, the developer has probably developed many applications and understands what is needed to make it. Therefore, next to the number of years of experience, the amount and type of applications developed influences a developer’s expertise level. Lastly, certifications can be used to measure the level of expertise (4 respondents). These certifications can be obtained by following low-code related courses or following company internal training that educates the developer on working within the company and security issues. According to the respondents, these team levels and the team’s roles determine the level of control that the central team should have over the federated teams.

*‘So it’s just over time once we see the experience of these people [developers in the federated team] are increasing, we can decrease the central or support that we provide to the team’ – Respondent A, Benelux Leader Automation Services, Technology consultancy*

To conclude, the influence that a central team should have over federated teams depends on four factors. The control of the central team may diminish as one or more of these factors change. The identified categories influence each other too. For example, low-mature teams will not start developing a complex, multi-layered, external-facing application with several API integrations to critical databases. Furthermore, the more dependencies a federated team has on the central team, the more strict control is needed.

The last two factors are dependent on the platform level and can change. Therefore, these two factors should be incorporated in the design of the governance model. To find out how central teams could govern teams on the level as described in *Definition 2*, we looked at what kind of tasks a central team can have and what mechanisms and structures can be used to achieve proper governance.

### **4.3.3 Governance tasks**

In a federated low-code model, central teams should execute several tasks. A set of mechanisms could support these tasks. Both are identified during the interviews with all stakeholders. The tasks are divided into eight categories which are described below and can be found in *Figure 15*. During the interviews, it is sought to find answers to mitigate the risks and limitations identified in *subsection 4.2.2*. In this subsection, these findings will be discussed.



Figure 15: Central tasks for governing low-code federated development teams



### *Defining a strategy, operating model & responsibilities*

During the interviews, it became clear that a central team should define the strategy and operating model. Interestingly, this is in line with the findings of Anagnoste (2013) of setting up a centre of excellence for RPA. As explained in subsection 4.2.1 and as elaborated on in subsection 4.3.2, the respondents emphasized different ways to set up an operating model, including different working methods. However, the central team should always have a strategy and a vision to focus on growing (5 respondents). This also includes change management and aligning essential stakeholders to ensure that the federated model is adopted and to place the low-code software in the wider IT ecosystem of the company. Transparency and communication could help to create a culture that nurtures change management in a company (Weill & Ross, 2004a). Furthermore, a central team could overview the whole portfolio of all applications built by the federated teams to review the health and the performance of the portfolio (4 respondents). Moreover, in line with the research of Weill & Ross (2004a), central support and sponsorship could also help with this to partially fund the federated teams to lower the barrier to start a federated team.

Moreover, since the central team is defining the strategies and the operating model, the federated teams should understand their responsibilities. A critical recurring topic was the team composition of federated teams. From the interviews, the federated teams should consist of multiple developer roles, with at least one expert developer (6 respondents). Moreover, the team should consist of testers (3 respondents), business analysts (2 respondents) and a product owner or tech lead (3 respondents). The respondents especially stressed that expert developers should be included to achieve quality. The starting developers could learn from the knowledge of the experienced developer(s). Next, as discussed in subsection 4.2.1, federated teams should also take care of the operations when a solution is developed. Therefore, federated teams will have responsibility for the full lifecycle of their applications.

*‘It starts with somebody with experience so you can build around that person’* – Respondent H, Chief Technology Officer, low-code vendor



### *Communicating and aligning*

Interestingly, similar to the findings of subsection 3.3.4 and the purpose of a federated model described in section 1.1, the interviewees emphasized that the central team should ensure that the federated teams are aligned. This can be achieved by, for example, implementing a community (9 respondents). According to the interviews, this allows organizations to bring the teams together and share knowledge. In addition, central teams could organize hackathons (2 respondents) or regular meetings (2 respondents) with all federated teams. However, the bigger a community gets, the more the community should be adjusted to the community's preferences. Another way to align federated teams is by having a similarly clear vision among the teams, which was in line with the findings of Williams & Karahanna (2013) and Lindstöm et al. (2017). These alignment mechanisms could mitigate the risk of the dysfunctional barrier, which is explained in subsection 3.3.1.



*'You need to align. But the thing is, organize around functional topics (so business functions and also around technical topics). That is something to always keep in mind.'* – Respondent H, Chief Technology Officer, low-code vendor

The central team should also communicate about the low-code platform to federated teams and other parts of the business. The communication goals to other parts of the business could promote the technology across the enterprise (5 respondents). This promotion should be focused on explaining the capabilities of the low-code platform, providing transparency in the responsibilities of the federated model, and supporting the platform's adoption. Mechanisms related to these communications on the platform could include announcements, presentations, updates, outages, developer-awards, displaying already built applications, or newly available training.

*'The central team, they also have a task of evangelizing the technology by making sure that the organization is aware of the capabilities and what can be achieved'* – Respondent M, Chief Technology Officer, low-code consultancy



### *Providing development guardrails and organizational policy*

According to the respondents, the central team should outline the development of guardrails and codes of conduct that the federated teams should adhere to (8 respondents). In addition, similar to what was found in *subsection 3.3.4*, the respondents point out that the deployment stage is crucial in the development process, which depends on the team and should be well defined by the central team. The development process and other processes such as the deployment process (7 respondents) should be clear to the federated teams. A mechanism to ensure the compliance of the guardrails is a manual, as explained in *'Knowledge sharing and educating'*. The central team could also set the testing framework for federated teams to set a quality policy (6 respondents). Furthermore, according to the respondents (5), the central team should set the architectural framework. Therefore, it is essential to have a solution architect in the central team that understands the best ways to lay the foundations for applications. This role can define the guardrails from a software architecture perspective.

*'I would recommend that the central team sets the basic rules and guidelines, and everyone should be using that. So the central team defines the means and ways, and the central team defines the rules and instructions'* – Respondent C, Head of Intelligent Automation CoE, Technology consultancy

In contrast to what was found in *subsection 3.3.4*, where the central team should execute prioritization and identification, the respondents mentioned that the federated team should fulfil these tasks. However, the central team should help the federated teams in identifying new opportunities.



### *Knowledge sharing and educating*

As discussed in *chapter 3* and in *'Communicating and aligning'*, the central team should ensure that the teams are aligned in a federated model. With this alignment, the central team should also ensure the distribution of knowledge across the federated teams and support knowledge sharing. All interviewees stated that there should be an explanation of either the way of working for federated teams or explaining the platform. As explained in *'Communicating and aligning'*, the central team should also communicate to the whole organization about the federated structure as well as the low-code platform to mitigate the risks identified in *subsection 4.2.2*. This also includes an explanation of the development process and the capabilities of LCAP (10 respondents). In this way, the central team educates the federated teams and the other parts of the organization to ensure the adoption of the platform and a better understanding of ways of working.

*'And I see a lot of organizations when they start with low code; they don't necessarily know yet how that's going to be for them. It's also a little bit experimental for most people'* – Respondent N, Senior solution architect, low-code vendor

As explained in *subsection 4.2.2*, certifications could be a measure of the maturity of a team. Therefore, federated team developers can be trained to receive certificates to mature as a team and build more complex applications. Mechanisms to achieve this are training (7 respondents), platform presentations and

a developers manual that explains all the codes of conduct and ways of working, which are defined in *'Providing development guardrails and organizational policy'* (3 respondents). This training could be based on improving the technical knowledge of the development, training focused on the way of working or internal security guidelines. Moreover, a certification program linked to permissions to build specific types of applications can be set up (2 respondents). In this way, maturity and autonomy can be earned by following courses.

*'You just want them to know how to build apps. So there needs to be an explanation to those Federated teams on how we develop, what the development rules are but also, the way of working'* – Respondent K, Manager, low-code consultancy

*'We need to constantly coach them and educate them how they keep on top of the things that should be done, or that should not be done'* – Respondent G, Global Senior Director, low-code vendor



### *Maintaining and managing reusable assets*

As elaborated in *section 3.2*, reusability is an essential topic within low-code. By aligning the federated teams, knowledge on the different reusable components can be widely spread. From the interviews, it became clear that managing and maintaining reusable components should be done by the central team (6 respondents). An example of this centrally governed component is the SSO component which is explained in *Footnote 7*. Next to these types of shared components, the central team should also offer out-of-the-box building blocks with integrated UI/UX branding (4 respondents). Therefore, a UI/UX role that focuses on the company's branding is needed in a central team. In this way, federated teams can speed up their development because they do not have to think about this.

*'Because there are templates available which speed up your process, and those may have UI/UX patterns and stuff like that. So, UX development is becoming more and more prototyped and templated as well'* – Respondent I, Chief Executive Officer, low-code vendor

Furthermore, when federated teams develop an application or component that could be useful for other teams, this must be standardized first by the central team (4 respondents). The central team should also identify which components could be helpful and are demanded by the federated teams. This identification is only possible if the central team has a good overview of all federated teams and aligns them. A mechanism to promote reused components is by showcasing them during regular meetings (5 respondents) with the community or having an app store explaining the developed applications or components (5 respondents).



### *Facilitating and maintaining technologies*

Similar to managing and providing reusable components such as SSO, the interviewees highlighted the task of being a platform owner. The platform owner should make sure that the platform works well and is always updated to the latest version and that the cloud infrastructure is in place to work appropriately (7 respondents). The platform owner should also act as a facilitator and make sure that the federated teams can develop autonomously. This means that the federated teams can work in a self-service way when having the authority to reuse a set of components without permission. This facilitator role also includes that the central team needs to ensure that the security is centrally managed (6 respondents). However, this again depends on the type of organization and application; if a federated team wants to use personal data, the general guidelines should be explained by the central team, but the laws on personal processing data could differ per country. In line with the findings of *section 3.3.1*, because the central team has a facilitator role, licenses should be managed centrally and distributed to members of the federated teams. All in all, the central team should automate the development process as much as possible to ensure the design freedom of the federated teams, as explained in *section 4.2* (6 respondents).

*'The central team is responsible for creating the environment in which the federated team can start building in an efficient way and start continuously improve in an efficient way'* – Respondent N, Senior Solution Architect, low-code vendor

As explained, many respondents (5) emphasized creating a process for deployment, and some suggested doing code reviews before going into production. As explained in *section 3.2* (nr 5), a CI/CD pipeline with built-in test frameworks that can do, e.g. regression testing, can help to automate the development process and is especially essential when the number of federated teams increases and when there is a high number of development lifecycles (3 respondents). A CI/CD pipeline should ensure that mature teams can develop without many barriers, whereas the starting federated teams that need more oversight and control are more guided.

*'You don't want to become the bottleneck as the central platform team. But you do want to have some control and quality. So that's that speed versus agility dilemma' – Respondent M, Chief Technology Officer, low-code consultancy*



### *Monitoring, assessing and evaluating*

According to some respondents (7), the central team should monitor the federated teams and the applications that they are going to build. However, the degree of monitoring depends on the factors presented in *subsection 4.3.2*. This reviewing could be done by mechanisms such as code reviews (7 respondents) or walk-along sessions. However, to evaluate, the central team must understand what kind of applications the federated teams will be built (7 respondents).

*'I would definitely require code reviews in situations when a team is in a learning curve. So either when they're new, or when you've assessed that they have specific things that they need to be working on, it can be very useful to have code reviews' – Respondent N, Senior Solution Architect, low-code vendor*

Furthermore, during the interviews, the deployment process was mentioned (7 respondents). However, even since federated teams are autonomous, there should be assessments to validate whether the teams process private data. This security control is in line with the compliance and security risks found in *3.2* and *4.2.2*. Mechanisms to mitigate this are to review every application with a security and risk assessment that checks what type of data will be used and by whom (6 respondents). In this way, risks per application can be identified and monitored accordingly by the central team (7 respondents). In this pre-deployment stage, the central team can conduct code reviews to evaluate the application developed by the federated team (6 respondents). In this stage, penetration tests (3 respondents), access tests (3 respondents), or the previously discussed risk assessment can be conducted to ensure quality.

*'Even in autonomous teams, you will have some guidelines and checklists that they have to follow before they go to production' – Respondent H, Chief Technology Officer, low-code vendor*

Furthermore, monitoring is vital to capture the value that is being generated by the federated teams. This can be used to drive adoption or to drive change management at important stakeholders (2 respondents). By creating a similar value framework that every federated team uses, a standardized value measuring can be conducted. Therefore, precise value monitoring can be done to measure the success of the federated low-code model. A mechanism that can be used is a dashboard tool that tracks the developments of the federated teams (3 respondents).

As discussed in *'Defining a strategy, operating model & responsibilities'*, since federated teams could consist of experienced developers, these levels should be assessed to evaluate if somebody is 'an experienced developer'. Therefore, a maturity and complexity model with the defined dependencies, as explained in *subsection 4.3.2*, could be a mechanism that endorses this. In addition, by monitoring teams, their progress and performance can be monitored. As a mechanism, a dashboard could be created to have an overview of all federated teams and their associated applications (4 respondents).

*'So I don't mean that they need to check the work all of the time. If a team can produce quality work, then you don't need to go and babysit them' - Respondent N, Senior solution architect, low-code vendor*



### Assisting and advising

As explained in *subsection 4.3.2*, federated teams will need more guidance at the beginning of projects. Thus, especially in the start phase of federated teams, the central team should ensure that the federated teams develop according to the specified way of working and within the development guardrails defined by the central team. Therefore, the central team should coach the federated teams to certify that the applications are developed according to the right measures. For example, as explained in *subsection 4.3.2*, the type of application could also influence a central team's impact. For example, when an application will be developed that could potentially be beneficial for other parts of the organization, a solution architect should be involved to build a solid foundation for the application. In these cases, the central team should have capabilities such as an architect to assist and coach federated teams with applications that require more attention to the application architecture.

*'And then when the experience of that federated team becomes higher, and the knowledge becomes higher, then they can build the foundation of the next app themselves' – Respondent K, Manager, low-code consultancy*

An identified mechanism is to have a single point of contact (SPOC) from the central team that support the federated teams (4 respondents). In this way, an expert from the central team is involved in the development of the federated team and can advise on the development. When the teams are new, the SPOC can assist them through the development process and show how the developments should be done. In addition, as discussed in *'Monitoring, assessing and evaluating'*, a central team should understand what federated teams are developing (7 respondents). When having this information upfront, the experts in a team can advise the teams on how to develop the application or advise what previously developed components they can reuse. Moreover, when the maturity of the federated team is determined, the central team can advise, for example, whether or not they can build specific applications. For example, based on the information, the central team can recommend starting with a smaller application or leaving out particular features in the first version. When having this SPOC, the central team will have a better view of the demand and difficulties of the federated teams, such as integrations to specific systems or training sessions on particular problems.

*'I would see the centralized team more in a coaching role of reaching out of being aware and more in a situation that people quite naturally reach out to them' – Respondent M, Chief Technology Officer, low-code consultancy*

In addition, the central team could advise the federated teams to work agile (5 respondents). However, since the teams can operate autonomously, the central team should not focus too much on the development method since that is a responsibility for the federated team itself. Again, this support by the federated team will decrease over time when the federated team gets more mature.

To conclude, during the coding process, some categories from **SRQ2** were adjusted to categories that were more suited in the context of low-code based on the conducted interviews. Eight categories of tasks were found that a central team must execute in a federated low-code model. Compared to the tasks identified in the previous chapter (see *Table 11*), managing developed assets was dropped. This was based on the results of the interviews and elaborated in *subsection 4.2.1*. Four other categories were renamed to categories that suit the identified tasks. One category, *'Assisting and advising'*, was added. These central tasks can be found in *Figure 15*, and each category was elaborated separately. The selection and design of these mechanisms will be discussed in *chapter 5*.

#### 4.3.4 Governance models

According to Weill & Ross (2004a), an operating model should be defined when defining a governance model. Moreover, as shown in *Table 9*, IT governance depends on five different IT decisions that should be clarified. The first decision depends on the IT principles, where a strategy and operating model is chosen. Moreover, as described in *subsection 4.3.2*, a governance model depends on the operating model. Therefore, a cross-case analysis was conducted among enterprises from various industries to find

similarities, best practices and processes in operating models of software developments. The interviews' output is described below, and the visualisation of the operating models, including additional information of the interviewed companies, can be found in *Appendix VI*. During the interviews, similarities were discussed to understand best practices of operating models in software development. It is hard to compare other companies' operating models since no company was interviewed with a similar federated low-code model. Either the federated model was more focused on citizen development, or another technology was used. In one case, only a central operating model was used. Below, the similarities are described.

### *Organizational constructions of bringing in an expert developer*

Interestingly, similar to what was found in most interviews and explained in '*Defining a strategy, operating model & responsibilities*', all case companies always had an expert developer in a way incorporated in the model. However, since the oil & gas company used citizen developers and a no-code platform, the construction of expert coaches were linked to all developers. In this construction, the coach is linked to one or more teams, and since they are closer to the business, they have a better understanding of the needs and can, therefore, respond accordingly. This construction can be linked to the SPOC mechanism.

*'They [the coaches] are better in touch with their business units, and generally, people in the centre simply don't know enough or about the problems going on in that specific area. So we have very wide range of teams within [company name], and our general delivery teams are very bad at knowing what the business wants. So having somebody business proximate who does know that but who also is closer to you if you want to ask questions, that's really helping them along, and they can link up with us if they want any additional information'* – Respondent D, DIY Implementation Lead, Oil & Gas

Moreover, when developing more significant projects (no citizen-developer projects), there are always multiple developers in the team that co-developed with experts from the central team. In addition, the banking II company used a centralized operating model with different types of developers. This is shown in *Appendix VI*. However, similar to the found risks in *subsection 4.2.2*, having an expert developer in a federated team is not always possible due to the higher costs of expert or full-stack developers. Therefore, as described, similar constructions could incorporate an expert level to a federated team. Furthermore, most companies also used third parties to be able to involve expert developers. This also enables companies to be more flexible. This is another type of construction that could be used to involve expert developers in the development of applications. However, these 3<sup>rd</sup> parties should focus on educating the starting developers to grow to be self-sustainable. This is in line with what respondent H mentioned:

*'What often happens in practice is that they don't have enough people in the federated team that are on the right level. So then they actually hire partners to do that to help kick start the projects, train their own people on the job and so after a while they can do it themselves'* – Respondent H, Chief Technology Officer, low-code vendor

Industry	Banking I	Oil & gas	Chemicals & logistics	Banking II
Answer	✓	~	✓	✓

### *Intake process*

All interviewed companies but one used an intake process before the start of the development. Only the oil & gas company did not make use of this. However, this was due to using a no-code technology platform where the company allowed citizen developers to build applications. With these software development platforms, it is easier to centrally control and define functions, which also ensures that fewer complex applications can be made. Furthermore, in the banking I company, the central team provided licenses to build a process based on the assessment outcome of the intake. From the information provided by the federated team, the central team decided whether the federated team was allowed to develop a particular project or not. Therefore, an intake mechanism can be used to understand projects that are going to be developed.

Industry	Banking I	Oil & gas	Chemicals & logistics	Banking II
Answer	✓	✗	✓	✓

#### *Development guardrails and education by the central team*

As discussed in *subsection 4.3.3*, the central team should ensure that all guidelines and codes of conduct that federated teams should obey are provided in a transparent and accessible way. In all companies, a manual with processes and codes of conduct is created where these rules are stated. In addition, the central teams also provided training to all federated teams or citizen developers. In the banking I company, there were different certifications for different roles. Therefore, the company made sure that a federated team consisted of a well-trained business analyst as well as a well-trained developer. During this training, also internal knowledge on the way of working was explained.

Industry	Banking I	Oil & gas	Chemicals & logistics	Banking II
Answer	✓	✓	✓	✓

#### *Central building capabilities*

All identified teams also had central building capabilities next to the federated teams that can develop applications or processes. This was implemented to develop more critical and high-risk applications, which is in line with the findings of *subsection 4.3.2*. In this way, these applications are built with the central expert developers. This emphasized the need for expert developers in the central team. Therefore, the projects were also categorized.

Similar to the findings of *subsection 4.3.3*, another development method can be selected depending on the application that a federated team has. For example, if a business-critical external-facing application that should have 24/7 support would be built, the central team could build this within the central team. Therefore, a strict separation between projects was determined since federated teams will not always have the capability to offer, e.g., 24/7 support.

*‘The problem is, if you want to assure the right quality, you need to have a certain capacity in order to control that. And so for instance, being 24/7 alert and within one or two hours understand and see the incident that is raising if things fall down’*  
– Respondent A, Benelux Leader of Automation Services, Technology consulting

The chemicals and logistics company used ramp-up teams. Based on the intake and the risk assessment, an agile team was gathered with capabilities from the central team. When an application was developed, there was a hand-over to another support & operations team. This construction was used since the company size was smaller than the other companies, the scale of complete federated teams could be achieved. Therefore, this operating model was different compared to other cases.

Furthermore, these central building capabilities also have their risks. For example, business units may find it easier to let the central team develop the application. Therefore, incentives could be used to ensure the adoption of the federated model (Weill & Ross, 2004a). This risk was also noticeable in the oil & gas company, as can be seen below.

*‘The fact that we have a very small IT delivery team, so if people don't want to do it themselves, we might not be able to do it right, or you have to wait for a very long time’* – Respondent D, DIY Implementation Lead, Oil & Gas

Industry	Banking I	Oil & gas	Chemicals & logistics	Banking II
Answer	✓	✓	~	✓*

*\*Company only has central development capabilities*

### Risk assessment

All companies conducted a risk assessment before development or used a model to categorize the type of projects. In the oil & gas company, a zoning model<sup>10</sup> was used to define the complexity and divide what roles can develop which part of the zone. This is in line with the mechanism of a maturity and complexity model where applications can be assigned to federated teams' maturity level. When the project was too complex, it would be built by the central team. In addition, banking II classifies their developments into different categories with corresponding processes.

Industry	Banking	Oil & gas	Chemicals & logistics	Banking II
Answer	✓	✓	✓	✓

To conclude, next to generic similarities such as an agile way of working, providing training, involving expert developers and having development guardrails defined by the central team, other similarities were found. Also, similar to mechanisms identified during the other interviews, the companies of the cross-case analysis also used mechanisms, e.g. internal communities, developers manuals, and an app store. As can be seen, all companies also had central building capabilities to create applications with a broader scope or that have higher risks. Moreover, various constructions to bring in expert developers in the development process were discovered. When looking at the development process, all companies conducted a risk assessment. In addition, three out of the four companies used an intake process to understand what the federated team wanted to build and advise on this. Based on the information of this chapter, a list of mechanisms is provided, followed by a mechanism selection and design explanation of the artefact. This can be found in the next chapter.

<sup>10</sup> More information on the zoning model and the review of the Oil & Gas company can be found in *Appendix VI*.

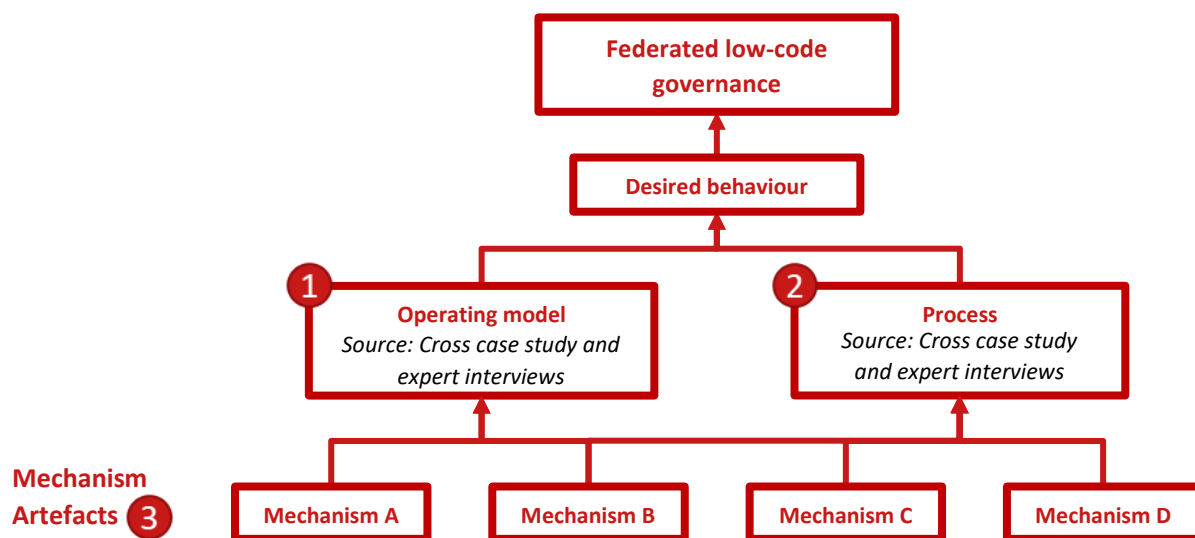


## 5. Artefact design

In this chapter, the designed artefacts are discussed. The artefacts were dependent on the case company's demand and depending on the case context. Therefore, an overview of the artefacts is presented in *section 5.1*. Thereafter, the case company and its context are described in *section 5.2*, followed by the mechanisms that could be implemented in such a governance structure (*section 5.3*). Lastly, the designed artefacts are discussed in *section 5.4*.

### 5.1 Artefacts overview

For this research, various artefacts were designed that originate from the explorative case study and literature study. Therefore, output from previous chapters is used to define and propose a governance model with mechanisms. The artefacts consist of three components where the last component consist of multiple mechanisms artefacts. This is explained in *Figure 16*, and they are briefly described below:



*Figure 16:* Artefact overview for governing federated low-code development teams

First (1), as noted in *section 3.3*, in addition to determining processes, governance is also involved with structures and operating models that must be established in the concept of IT principles (Weill & Ross, 2004a).

Next (2), from the interviews, it became clear that explaining the development process or platform was one of the most cited tasks by the interviewees (10 respondents). In addition, to ensure that the federated teams follow the desired behaviour, a process should be determined, which is also an element of governance. Therefore, a federated development process for starting federated teams is developed according to the outputs of *chapter 4*.

Lastly (3), since a set of mechanisms and tasks is found in *chapter 4*, these can be designed and implemented in the case study context. The alignment between the central entity and the federated teams could be improved by introducing mechanisms (Brown, 1999). Based on the case understanding, implementation difficulty, and amount of references from *chapter 4*, a selection is made to which mechanisms are developed. These designed mechanism artefacts will later be demonstrated and evaluated in the case company.



## 5.2 Problem understanding context

In *section 1.2*, the case company is already briefly introduced, and an analysis of their original process and operating model can be found in *Appendix III*. This central team consist of a product owner, expert developers, an architect and UI/UX designers. Furthermore, the team works closely together with other Intelligent Automation teams, as was shown in *Figure 6*, and the central integrations team that takes care of its information systems' integrations. The intelligent automation department also consists of a central security officer that manages the security and ensures that the teams are compliant with the security guidelines. In addition, a developers manual is defined with technical guidelines to make sure that the developers adhere to the codes of conduct. Furthermore, similar to the interviewed companies in *subsection 4.3.4*, the central team also has building capabilities to develop applications.







### 5.2.1 Problems




Similar to the risks identified in *section 3.2 and subsection 4.2.2*, the case company also confirmed the identified security and compliance risks. As explained in *section 5.2*, the current process is not scalable; when the number of federated teams increases, there is no clear overview and capacity to monitor and assess all applications. Furthermore, the central team can ensure that the applications they develop in-house have sufficient quality because they follow a specific process. However, they cannot make sure that the federated teams also will deliver high-quality applications. This is also because the central team understands the development process, while this might not be the case in the federated teams. Therefore, this resulted in having manual code reviews to ensure that a quality level is achieved, which is not scalable when increasing the number of federated teams. This is in line with the lack of expertise and the adoption of new ways of working risks, as found in *subsection 4.2.2* during the interviews with external interviewees.

## 5.3 Selecting mechanisms

Based on the analysis of the case company in *section 5.2*, it is found that some mechanisms were already implemented, e.g. a SPOC from the central team, central building capabilities, code reviews, and defined codes of conduct in a manual. Therefore, a selection of possible mechanisms described in *Table 15* could be developed and implemented in the case company. The identified mechanisms originate either from the interviews in the explorative case study of *chapter 4* or from the literature study in *chapter 3*. The mechanisms found in the explorative case study can be traced back to the axial codes, which can be found in *Appendix IX*, or to the literature study, which can be found in *chapter 3*. Some mechanisms could support multiple categories, e.g. a development manual will educate federated teams and support adherence to the development guardrails. Furthermore, some mechanisms required too much technical knowledge, e.g. implementing a CI/CD pipeline or developing a test framework. Therefore, these mechanisms will not be reviewed.

**Table 15:** Mechanisms in a federated low-code model

 Category	Mechanisms linked to a task category	Source
 Defining a strategy, operating model & responsibilities	Involvement of senior management, change-management program	Interviews, literature study
 Communicating and aligning	An internal community, a SPOC from the central team, regular meetings, hackathons, awards, outages automation, announcements	Interviews, literature study
 Providing development guardrails and organizational policy	Development manual, explanation and transparency of the development process, security assessment, define codes of conduct, developers manual, training	Interviews, literature study
 Knowledge sharing and educating	Training (internal process & technology), hackathons, certificate program, explanation and transparency of the development process, security assessment, an internal community, development manual	Interviews, literature study
 Maintaining and managing reusable assets	An app store, an intake process, an internal community	Interviews

	Facilitating and maintaining technologies	Central building capabilities, experts in the central team, CI/CD pipeline, security assessments	Interviews
	Monitoring, assessing and evaluating	A dashboard, maturity and complexity framework, certificate program, code reviews, security assessment, an intake process, walk along sessions, spot checks, PEN-tests	Interviews, literature study
	Assisting and advising	A SPOC from the central team, an intake process, maturity and complexity framework, a test framework, explanation and transparency of the development process	Interviews, literature study

#### A. An internal community

When building an internal community among all federated teams, they can meet and learn from each other. An implementation is relatively easy and can be seen as a quick win. Furthermore, hackathons and regular community meetings could increase informal contact among federated teams, increasing the knowledge sharing among each other (Tsai, 2002). In addition, building an internal community could also promote the low-code federated program within organizations and the visibility and knowledge of low-code development among colleagues. Moreover, when federated teams feel part of the same community, it can ensure adherence to these codes of conduct. Therefore, it is crucial to gather all teams in one place. The central team should operate as a moderator in these community sessions to align the teams to nurture the social aspects of the community and focus on learning (Gray, 2004). Nevertheless, a community should be maintained to work effectively and be more beneficial when the number of teams increases.

An internal community with weekly interdepartmental presentations or workshops events can be seen as an informal mechanism (Brown, 1999). Drawing upon previous research on mechanisms in IS organizations, implementing such an informal mechanism can be classified as low-implementation costs (Brown, 1999). Therefore, it is assumed that this cost aspect is also lowering the difficulty barrier for implementation because less political support needs to be created for the budget within a company to implement the mechanism. Therefore this mechanism will be implemented.

#### B. Maturity and complexity framework

As discussed in *subsection 4.3.2*, the governance of federated low-code teams, and therefore the desired behaviour, differs per use case. Since in low-code, the experience of developers can vary significantly, this difference is more substantial. Therefore, similar to case company B, a zoning model, as discussed in *section 4.3.4*, could help the enterprise categorize projects and teams with matching governance processes and structures. In *subsection 4.3.2*, a start of this framework is provided. However, this should be reviewed more broadly and validated with various stakeholders. In addition, different measures per development process have to be added accordingly. There would not be sufficient time to research this and develop this mechanism, and it will not be developed.

#### C. Certification program

As explained in *subsection 4.3.3*, a certification program could provide developers insights into their learning path to build specific applications. This certification program could be complementary to the maturity matrix as explained in *mechanism B*. This will give the federated team a timeline to develop e.g. applications with connectors of level X or type of data of level Y. Since this mechanism requires more in-depth knowledge of the types of projects and corresponding required role levels, it will not be developed.

#### D. Education, training and a manual

As discussed in *section 4.3*, educating the federated teams and explaining the functionalities and the desired behaviour was one of the most recurring themes during the interviews. Therefore, after the desired behaviour was determined, explanations on the process and platform should be provided. To ensure this desired behaviour of federated teams, a manual could be created, or training can be provided. Moreover, internal training on this process, security guidelines and branding can be provided. Since the education mechanism was a broad scope, there was a focus on creating an explanation of the development process which has been developed.

### E. Security self-assessment

During the explorative interviews with other companies and interviewees, it was found that a risk or security assessment was commonly used. Since we have found that security and compliance issues were more significant risks in federated low-code teams than central development, a security self-assessment could mitigate this risk because it allows federated teams to assess their application regarding the security and risks themselves. Therefore, this mechanism automates the development process and contributes to the freedom of design feeling in federated teams. When having central monitoring of these assessments, there can be more guidance and control based on the risk level involved. Therefore, not all applications need to have a manual review and meeting with the security officer. Since the security information was available within the company to create this, security was part of the artefact requirements, and it covers many tasks, this mechanism will be developed.

### F. Intake process

As was found in *chapter 4*, one task for the central team is to understand the applications that the federated teams will build. When having this knowledge in place, it is easier to have a general overview, assist and advise them on the development, architecture, identify risks and support reusability. This is in line with the example of the operating model of ING, as presented in the study by Weill & Ross (2004a). Moreover, it gives the central team the option to measure the value of the federated teams if all products are submitted via one central channel. This intake procedure was also in place in most other case companies studied in the previous chapter. Therefore, this mechanism was designed and incorporated into the development process.

### G. Dashboard

A dashboard ensures a central monitoring capability for federated team progress. Tracking applications are, most of the time, already built-in in leading LCAP. However, tracking which applications have a high risk or process personal data should be monitored too. In addition, a dashboard also helps measure the federated program's success, which was also indicated as an essential task for the central team in *subsection 4.3.3*. By monitoring, the added value can be measured that the portfolio as a whole adds. In this way, a central team can provide insights such as the number of teams, amount of projects and types of projects, too, for example, a Chief Information Officer (CIO) or Chief Financial Officer (CFO). In this way, the results of the federated low-code program can be communicated.


















*'So the more responsibility you have as a centralized team, the more monitoring you need to do'— Respondent K, Manager, low-code consultancy*

### H. An app store

Lastly, as discussed in *chapter 3*, an app store filled with applications and components developed by the central team and federated teams will help in various ways. First, it shows what is available and developed within the company to stimulate reusability among federated teams. Secondly, it will inspire federated teams to develop. Lastly, it promotes other parts of the enterprise since it shows what LCAP can do within their business. In this way, it supports the federated program within the company. However, since most LCAP offer similar mechanisms, there was no focus on this mechanism.

To evaluate what mechanisms can be developed that can be used to support the governance of this federated model in the case study, internal meetings with the case company are set up to review the demand and problems. Based on the company's demand, the feasibility, and the categories that the mechanisms cover, four mechanisms are selected to design as an artefact. This study will simultaneously implement multiple mechanisms since assessing a set of mechanisms could be more effective (Brown, 1999). As described in *section 1.3*, artefact requirements drive the artefact's design and satisfy the stakeholder goals. In *Table 16*, also the tasks of categories that these mechanisms intend to cover are shown.

**Table 16:** Selected mechanisms for the study that were designed

Mechanism artefact	Description	Covering tasks and responsibilities	SG & Req.*
	Explanation and transparency of the development process	Define a process and create an internal online channel with explanations on this complementary to the existing manual and create explanatory information for the federated teams	   A, B, D, E, H
	An intake process	Develop a process for federated teams and other colleagues to submit their ideas to advise and assist them in the development	   A, E, G, H
	An internal community	Create an internal community with regular meetings for presentations and one channel where the teams come together	   B, C, E, G, H
	Security self-assessment	A security self-assessment that allows the federated team to assess the security level of the application and to alert the measures that should be taken accordingly	    E, F, H

\* Pre-defined stakeholder goals (SG) and requirements (Req.) from Table 3

## 5.4 Artefacts designs

As discussed in *section 5.1* and shown in *Figure 16*, various artefacts are developed; the four selected mechanism artefacts of *Table 16* (discussed in *subsection 5.4.2*), a development process (explained in *subsection 5.4.3*), and an operating model (shown in *subsection 5.4.1*). These artefacts combined with the tasks shown in *Figure 15* form a holistic approach to govern federated low-code teams.

The proposed operating model and development process originated from the explorative case study from *chapter 4*. The designed mechanism artefacts are developed to support this proposed model. These designed mechanism artefacts are more tangible designs that can be evaluated. To better understand how the identified categories from *Figure 15* and mechanism artefacts, shown in *Table 16*, relate to the proposed process and governance structure, they are plotted on both figures. The categories are indicated with a circle and its corresponding icon. The designed mechanism artefacts are shown as a rectangle.

### 5.4.1 Implementation of mechanism artefacts for federated low-code governance

All implementable mechanisms were discussed in *section 5.3*, and the way the researcher implemented the four selected mechanism artefacts within the case company is briefly elaborated below (3 in *Figure 16*). It is discussed how the artefacts are designed and what the underlying goal was. More information on the designed artefacts can be found in *Appendix X*.



#### *Explanation and transparency of the development process*

To make sure the federated teams work according to the desired behaviour, *Figure 17* is adjusted to the case company. The process is presented in a digital open intranet page that the federated teams can access to understand how the process works. The process is created interactively; by clicking on the stage or check, the user will be redirected to another page with more extensive information. In this way, the user is guided through the process and educated along the way. This artefact aims to reduce manual work by the central team since the federated team can guide themselves and learn from the explanations. More information on the design of this mechanism artefact can be found in *Appendix X-A*.



#### *An intake process*

Because of the central team's central building capabilities, the intake process was also used to track, monitor, and assess demand for this development type. As explained, the goal of an intake process is to get an understanding of what the federated team, or in a central development case, wants to develop. By

having this understanding, the central team can advise and help the requester on the development or advise if it is feasible to build. Moreover, the central department would be able to advise what common components could be used since they will have a central overview and maintain the central repository. In this way, the mechanism would support reusability within the company.

Since the case study consists of a department with two other technologies (RPA and no-code), questions were formulated more broadly. Team members of these teams analyzed the intakes since the request that comes from the business could also have a better fit to be solved with that technology compared to low-code. Since the intake process will analyze intakes instead of setting up meetings, manual work would be reduced. The central security officer, which covers these three functions, was also asked to add questions for the intake process.

This artefact was developed iteratively and was designed with a no-code tool where multiple databases and web tools are linked to one and another. Moreover, a dashboard was created to generate a user-friendly general overview to handle and monitor incoming intakes. This also allowed the central team to measure the number of incoming intakes to measure the federated program's success, as was suggested during the interviews in *chapter 3*. More information on this part of the artefact can be found in *Appendix X-B*.



### *An internal community*

An internal community was organised within the case company. The goal was to align the federated teams by creating one community channel and providing all platform-related news, updates, and information in this channel. In addition, regular community meetings were conducted where topics were explained, and the teams were brought together. Therefore, a list with topics that could be presented to the federated teams was defined beforehand. This list was determined by asking questions regarding the federated teams' demand and the need for explanations of the central team. Furthermore, during the community meetings that were organized, the process was explained and other topics such as the implementation of SSO internally (*Footnote 7*). These common components and presentations about developed products by federated teams were organized to support reusability and inspire federated teams to build new applications.

Furthermore, these meetings were recorded and stored on an intranet location that the federated teams could access. Having these presentations stored locally, manual work would be reduced when new federated teams join since they could be redirected to the created presentation. More on this mechanism artefact and its implementation can be seen in *Appendix X-C*.



### *Security self-assessment*

As explained in *section 5.2*, the central team also experienced security and compliance risks with federated developments. In the case company, the security officer assessed the safety of the developed applications by filling in a manual Excel document containing the information classification of that particular application combined with a planned meeting. In their current process, the central team appoints a SPOC to a federated team to support. Before deployment, the central team always conducted a code review to make sure there were no errors.

The security self-assessment design started with designing a matrix of measures that should be taken into account by the federated teams when developing applications. Therefore, several meetings with two security experts from the case company were conducted. This matrix originated from the existing Information Security Procedure (ISP) developed for the low-code platform within the case company. However, this large security document is based on the platform level. Based on the ISP, information was extracted that was applicable on the application level and was later divided into three different levels; low, medium and high risks (as shown in *Table 25* in *Appendix X*). These levels were linked to a set of measures that correspond to the assigned level on a particular item (as shown in *Table 26*). For example, when having

the type of data on level X, measures Y should be taken into account. From this matrix, a security self-assessment was developed by the researcher with a no-code software tool. In this way, the federated team could fill in the assessment and understand per application what security measures should be conducted. Therefore, this mechanism would improve the security of federated developments. In addition, since the federated teams can conduct this self-assessment, manual work would be reduced because there did not have to be an intervention of the central team. More information on this designed artefact can be found in *Appendix X-D*.

#### 5.4.2 The development process for a federated low-code development model

The designed development process is shown in *Figure 17*. As can be seen, the process is divided into seven stages that are part of 3 phases. The phases are similar to the colours of the software development process by Brummelen & Slenders (2019), as was shown in *Figure 11*. The phases are divided into three phases to make a separation between the parties involved in the process; the central team, the federated team, or both. Furthermore, as explained in *subsection 4.2.1*, the federated team should use DevOps teams that take care of operations and undergo multiple iterations in the development process. Therefore, the iterative icon is shown in the right upper corner. Moreover, as can be seen in the left upper corner in *Figure 17*, this process can be used to explain to the federated teams what the desired behaviour is (☑). Furthermore, the process incorporated two checks where the central team will provide a go or no-go decision. As elaborated, it depends on the maturity and type of application how these checks look like. As shown in *A* (*Figure 17*), there are different processes allowed to ensure design freedom in the federated teams; less formal control from the central teams is needed for the more mature teams. This is in line with the leadership principles defined by Weil & Ross (Weill & Ross, 2004a), as discussed in *Table 10*. Moreover, it should be the task of the central team to assess and evaluate the federated (🚫). The mechanism framework, as discussed in *section 5.3-B*, would define when another process can be used. Depending on the team's capabilities, the development process can be adjusted, and more autonomy can be provided. Therefore, similar to the research of Williams and Karahanna (2013), static governance is shifted to the process of governing. Below each stage within the development process will be shortly explained.

*'Give people more autonomy where possible. So if you have that trade-off; choose autonomy where possible and foster the capability for those people to actually do that'* – Respondent N, Senior solution architect, low-code vendor

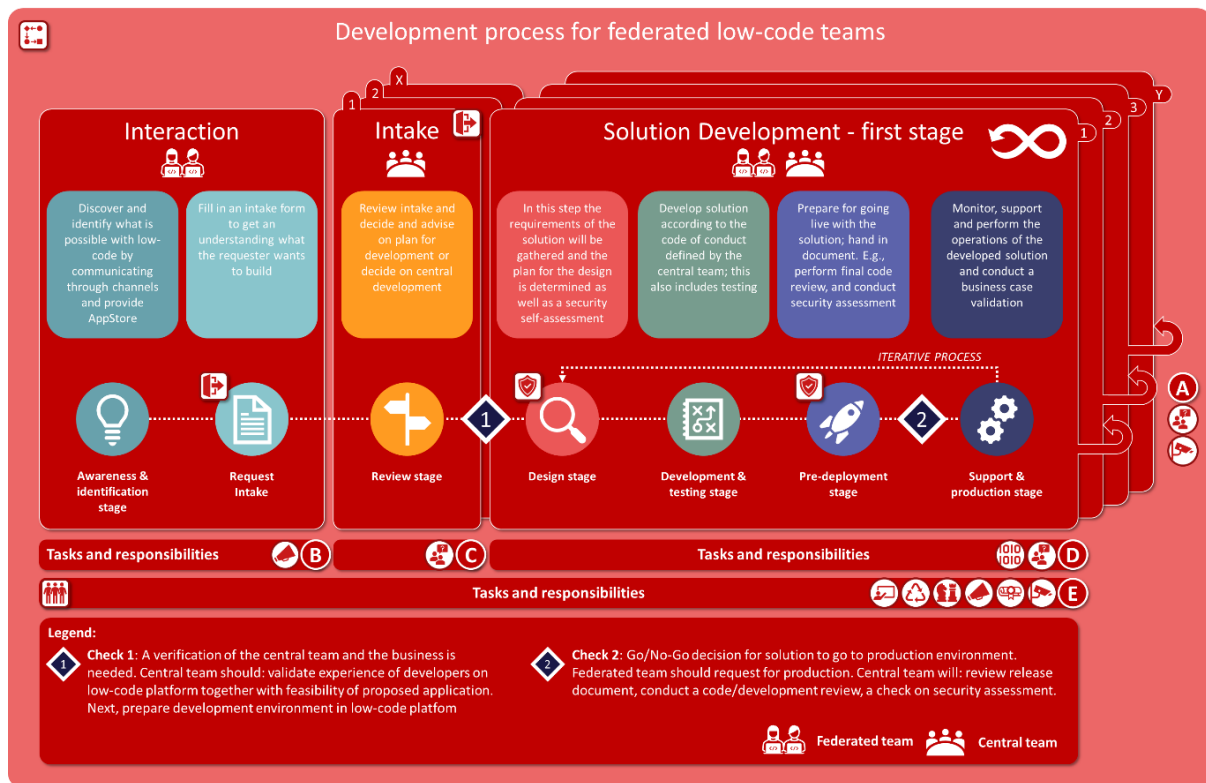


Figure 17: Solution development process for federated low-code teams

### Identification & Awareness

In the first stage of the process, a federated team will identify what they will build. A central team can help them by providing a manual to identify opportunities or to show pre-built applications. In addition, as discussed in *chapter 4*, the central team should promote their team and the technical capabilities of the platform to make the entire enterprise aware of the platform. Federated governance structures allow business units to identify opportunities locally. By placing responsibility and decision making on a local level, initiatives and prioritization could take place, and fewer political decisions have to be made. However, showing examples of developed applications and platform features is essential to get the most out of the federated team.

*‘The people that are working locally know the best what they use cases are, but sometimes you need to have, let’s say a change on mindsets and just see some examples from other domains to really be efficient and find the right opportunities’ – Respondent A*

### Intake

Similar to the other companies studied in *subsection 4.3.4*, an intake process is designed. Before federated teams start developing, they should fill in an intake form. This intake process allows them to submit information about the application to be developed. This intake can be filled in without any intervention from the central team. By documenting everything instead of having manual meetings, the intakes can be assessed offline by the central team by multiple people.

### Review stage

When looking at the review and completeness check, the central team should determine if a project should and could be executed, which depends on the factors as described in *subsection 4.3.2*. The level of questions asked also depend on these factors. For example, when a federated team has a high maturity level, the central should only know what is being built to align the federated teams to distribute knowledge

and possible applications to other teams. Therefore, an in-depth understanding is not needed because it does not have to be guided and coached by the central team. Furthermore, since the central team has an overarching role and monitors the developments, they can advise which intakes can use already developed applications or components better.

### **Check 1**

After the review, it should be decided whether the application that will be built should use pre-developed components, decide on the architecture and whether it should be built by low-code at all. This also emphasizes the need for expert developers in the central team. Therefore, this decision also depends on the type of application and maturity of the team. Before starting the development process, this should be validated by the central team. However, since the federated teams are autonomous teams, the central team should predominantly advise the team on their development. Though, for starting teams, the central team should have more control and, therefore, should provide a 'go' before starting the development phase.

*'So if we're talking this very small Excel replacement project, right? I think the centre of excellence doesn't necessarily need to be involved unless it's the very first project that the team does'* – Respondent N, Senior solution architect, low-code vendor

### **Design stage**

The first stage of the development phase is defining the user stories that will create the application and define the application's architecture. Since the federated teams often do not have an architect role, this central role should advise them on this. In addition, security has to be taken into account from the beginning of a project. Therefore, before starting the development, the federated teams should understand what security measures they have to implement. Therefore, the security self-assessment will ensure this.

*'You have responsibilities centrally in order to make sure that it's also from security aspect properly designed that people locally cannot be blamed later on that they have developed something because they didn't know about security procedures'* – Respondent A, Benelux Leader Automation Services, Technology consultancy

### **Development and testing stage**

This stage is where the actual development happens. During the development, tests should be conducted to verify if the design conforms to the business's requirements and codes of conduct. This can be done in the development, test and acceptance environment through beta testing, user tests, and automatic technical platform tests. This stage is in itself already highly iterative.

*'Then we go to the agile development side where we do planning, building testing, demoing and feedback, and that's all in the loop. A lot of our customers we are doing this with functionality by functionality, or maybe even user story by user story.'* – Respondent M, Manager, low-code consultancy

### **Pre-deployment stage**

Before deploying the first significant version of an application to the production environment, a beginning federated team should hand in documents to the central team, such as a verification of the security officer and test results of the application. The types and amount of documents could differ per enterprise, team and application. A company should clearly describe what documents are needed in what cases to ensure design freedom within the federated teams.

*'The centre of excellence needs to be involved at the beginning of the project and also at strategic points during that project just to keep, basically, some quality there'* – Respondent N, Senior solution architect, low-code vendor



## 2 *Check 2*


During this stage, the central team will review the documents that are handed in by the federated team in the previous stage. During this check, acceptance tests (UAT) should be verified. A final security check will validate if the application is developed according to the security guidelines, and code reviews by the central team should ensure quality within the application. This should be a formal approval in the first teams. However, depending on the application and features added, the federated teams themselves could do this. So especially in the first iterations, the central team should review the application.

*'Certain parts of their data, or certain attributes of an entity, they can be used by every citizen developer, for example, in the company. Other parts, maybe only after an approval, or only if their application runs within the firewall etc. So, you can always define all those rules think, that is where we should go to.'* – Respondent H, Chief Technology Officer, low-code vendor

During the interviews with external companies and the case company, this appeared to be a crucial point where the central team should conduct some tests before deploying, e.g. penetration tests (3 respondents), access tests (2 respondents) and security tests (3 respondents).

## *Support & production stage*

In this stage, the application is deployed in the production environment, and the federated team should monitor if any errors are occurring. If any errors arise or features have to be added, the team should go back to an earlier stage in this phase. As discussed in *chapter 3*, projects should be reviewed to measure the success or failure of a project after the first deployment (Prikladnicki & Yamaguti, 2004). By having all failures and successes in place, the low-code federated program's value can be defined more clearly, as was mentioned in *section 4.3*. It should be validated if the use case was successful, and this knowledge should be shared with other federated teams.

To conclude, based on the walk along sessions by the SPOC, the code-reviews during the pre-deployment stage, the teams and the quality of applications can be assessed. Until the central team achieves a level of trust, the central team can decide to move to another process where fewer documents have to move to a procedure that focuses on automatic deployment ().

### 5.4.3 Operating model for developing federated low-code

Based on the output of *chapter 3* and *chapter 4*, a federated low-code operating model was proposed which can be found in *Figure 18*. Below, the proposed operating model is briefly discussed. Similar to the development process of *section 5.4.2*, the identified tasks of *Figure 15*, are plotted on the operating model. In this way, a more holistic understanding of the artefact composition is visualized and shows how the artefacts relate to each other. The developed mechanism artefacts are presented in a rectangle shape and the identified tasks are indicated with a circle.

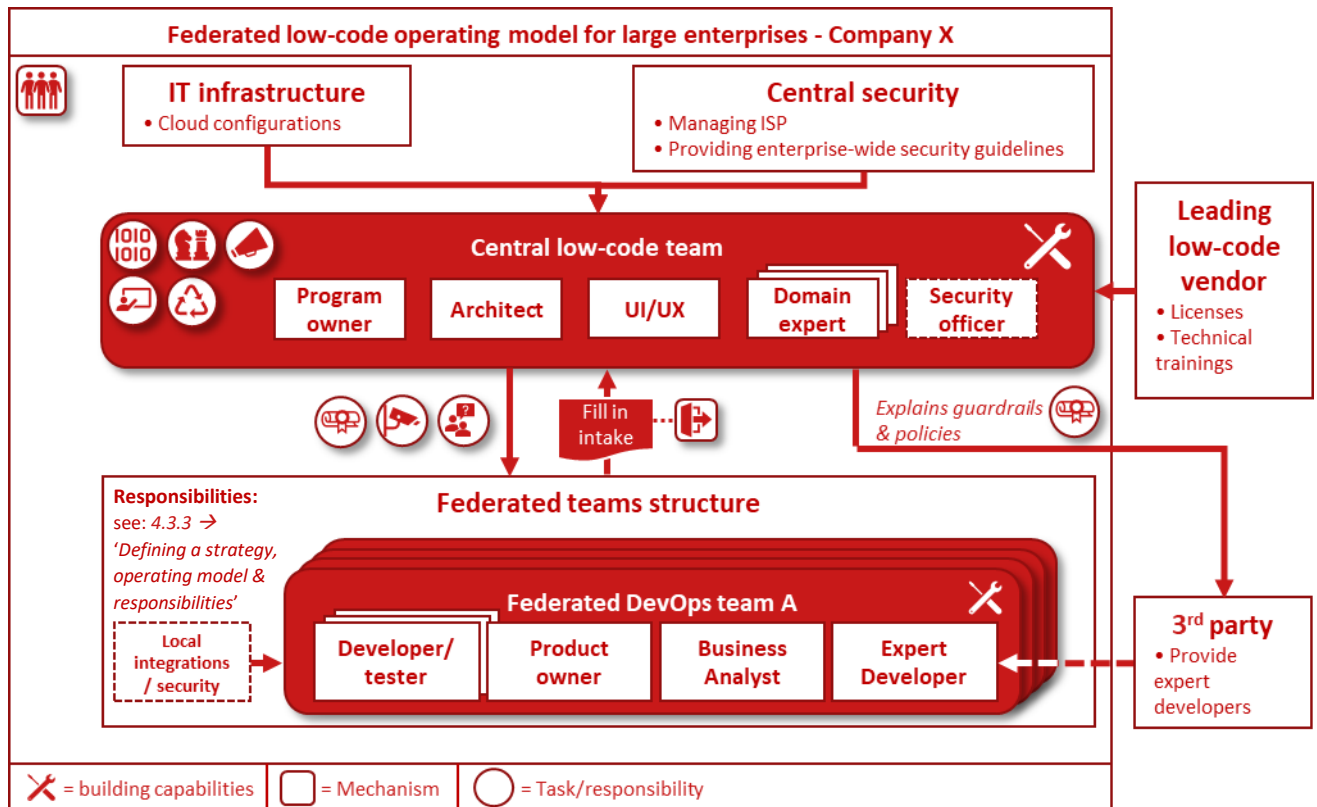


Figure 18: Operating model federated low-code model

As discussed in *chapter 4*, the federated teams should consist of multiple people. In *Figure 18*, the rectangles indicate roles, e.g., the product owner could also have a business analyst role. However, there should be at least two developer roles, including an expert. This expert-level can also be based on a third party. However, this expert knowledge should be implemented in the federated team in some way. Therefore, the central team should overview the federated teams that use third-party suppliers of expert developers to get into contact and explain the development guardrails and policies. The product owner is a role that should have technical knowledge yet has a managing role within the team.

Security guardrails and policies should be set up centrally according to the enterprise-wide regulations. However, when an enterprise is geographically dispersed, more specific local security and privacy-related topics may differ. Therefore, the first line of security measures should be set centrally, and when more local knowledge is needed and available, a regional security officer should be involved. Nevertheless, this is optional and is therefore depicted in *Figure 18* with a dotted line. Moreover, IT infrastructure is often managed by one central team that organizes all cloud control for the whole enterprise. Similar to local security, managing integrations with local IT systems could also be covered locally. Therefore this is shown as a separate entity. Furthermore, similar to *Figure 17*, a submitted intake is required by the federated teams.



## 6. Demonstration & Evaluation

In this chapter, it is discussed how the developed artefacts were demonstrated in the case company and subsequently evaluated. In *section 6.1*, it is explained how the designed artefacts were adjusted to and implemented in the case environment. Thereafter, the artefacts were evaluated to validate if they support the pre-defined requirements and validate the effects in the case context. First, the data collection set-up is described in *subsection 6.2.1*. Then, the artefact requirements are evaluated in *subsection 6.2.2* and the effects of the artefacts are presented in *subsection 6.2.2*. Lastly, the conclusions are presented in *section 6.3*

### 6.1 Artefact demonstration

The demonstration and evaluation are the fourth and the fifth step in the DSRM process (Peppers et al., 2007). For this study, only one case company was selected to demonstrate the designed artefact, which is described in *section 1.2*. Furthermore, during the development of all mechanism artefacts, several exploratory focus groups and interviews within the case company were conducted to gather information to incrementally improve the artefact (Tremblay et al., 2010).

#### 6.1.1 Design of artefacts adjusted to the case company

An adjusted development process was presented in the case company, which can be found in *Figure 19*. For the designs, the internal case company requirements were taken into account, shown in *Appendix IV*. This process included three new stages; detailed intake, the IT readiness stage, and the charging process. Moreover, another check was added. These additional stages and check were added to make the process more detailed and suited to the case company. For example, the charging process is a crucial factor for federated teams. Since this process was also used to explain how the development process worked and what the responsibilities of the federated teams were, charging was incorporated into the process (mechanism artefact A as described in *subsection 5.4.1*). In addition, as can be seen in *Figure 19*, this process is only based on beginning federated teams since there is only one level of the solution development is shown.

*-This figure is removed due to confidentiality reasons-*

*Figure 19: Federated team development process for starting federated low-code teams adjusted to the case company*

During the artefact design of the intake process, evaluation sessions were conducted where several intake questions were added or adjusted, or the process was transformed. Additionally, more generic questions are added to ensure that the intake process will cover the three teams of the intelligent automation department. In this way, the intake process enables the central team to handle the coming intakes with the right technology. Moreover, the researcher organized an alignment presentation for all stakeholders in which the intake process was explained. During this meeting, the artefact was shown, and feedback was gathered. The presentation is not added, however, the process that was developed in parallel to the intake process how it should be handled was explained and worked out in more detail, as can be seen in *Figure 55*. It was explained by the researcher how intakes should be handled and how they could be monitored with the dashboard attached to it. This new way of working was implemented and evaluated.

The security assessment was implemented in the development process and explained in the manual. A federated team from France that started developing used the security assessment. In addition, the assessment was verified and checked with the security officer. The communities were implemented and validated with both the central and the federated team.

The operating model of the case company was reviewed and visualized and can be seen in *Appendix III*. The proposed operating model from *Figure 18* was broadly similar to the existing operating model from the case company. Therefore, this artefact could not be evaluated except for the mechanisms that were plotted in the figure and the intake procedure.

## 6.2 Artefacts evaluations

According to Peffers et al. (2007), the evaluation should identify how well the developed artefact(s) work. As discussed in *subsection 2.2*, confirmatory focus groups were used to evaluate the artefact (Tremblay et al., 2010). To have a reasonable evaluation, the artefacts were implemented, used and later evaluated. The set of developed mechanism covered most of the found tasks and responsibilities as found in *Figure 15*. It was evaluated if each mechanism contributed in its own way to the pre-defined requirements as determined in *Table 3* and what the effects were in the case company. In addition, the effects of the artefacts were rigorously evaluated on three criteria (Hevner et al., 2004); quality, utility and efficacy, as discussed in *chapter 2*.

### 6.2.1 Evaluation focus group setups

As discussed in *subsection 2.2*, the focus groups were structured according to a predefined method (shown in *Table 5*). The focus groups were held with both the federated team as well as the central security officer and team members from the central team. The two focus groups should only contain participants that are familiar with the designed artefacts (Tremblay et al., 2010). Therefore, only stakeholders that used the developed mechanism artefacts were taking place in the two different focus groups, which are shown in *Table 17*.

**Table 17:** Participants focus groups








Participants	Function	Mechanism artefact*	Group A	Group A
<b>Participant 1</b>	Product owner RPA – Intelligent Automation	A,B,C	X	
<b>Participant 2</b>	Product owner no-code – Intelligent Automation	A,B,C	X	
<b>Participant 3</b>	Solution Architect Intelligent Automation	A,B,C	X	
<b>Participant 4</b>	Product owner low-code – Intelligent Automation	A,B,C	X	
<b>Participant 5</b>	Federated team developer 1	A,C,D		X
<b>Participant 6</b>	Federated team developer 2	A,C,D		X
<b>Participant 7</b>	Central security officer	A,C,D		X

\*Mechanism A: *Explanation and transparency of the development process*, Mechanism B: *An intake process*, Mechanism C: *An internal community*, Mechanism D: *Security self-assessment*

### 6.2.2 Artefact requirements evaluation

As discussed in *section 5.3*, each mechanism artefact contributed to a different set of artefact requirements. Independently, the artefacts did not fulfil all requirements. Nevertheless, combined they mark each box, as shown in *Table 20*. The results of the artefact requirements will be discussed below.

**Table 18:** Evaluation for artefact requirements and mechanism artefact

Artefact requirements (see <i>Table 3</i> )	Mechanism artefacts			
	 (A) Explanation and transparency of the development process	 (B) An intake process	 (C) An internal community	 (D) Security self-assessment
<b>F</b> It should help or support for a design for security	N.E.	N.E.	N.E.	
<b>G</b> It should help or support for a design for reusability	N.E.			N.E.

H	It should be designed in a way that manual work is reduced	✓	✗	~	✓
---	--	---	---	---	---

(N.E.) Not Evaluated

The explanation and transparency of the development process were developed on top of the proposed process of *Figure 19*. Therefore, this artefact was complementary and aimed to explain what to do at each development stage to reduce the work of the central team by documenting the development process. Having the explanations complementary to the development manual and using it during onboarding sessions, helped the central team to explain what the development process was. Additionally, the federated team appeared to use it to find how to develop and how the deploying process worked. Therefore, this requirement has been achieved.

The intake process (B), was designed to understand what the federated teams were going to build to advise on the type of technology, development, and to propose to reuse components. By advising before the development starts, and by receiving intakes via one channel could reduce manual work. However, the intake appeared to be useful for the central development intakes which led to an increase in the amount of work for the central team. Moreover, the intake partially supported reusability since the intakes gave an idea of the product that would be developed. However, still, a meeting was required to have an in-depth review of the request. Therefore, the intake served as the first line of analysis. Consequently, this requirement was partially fulfilled.

The regular community meetings component of the internal community (C) appeared to help align federated teams. For example, the developers mentioned that during these meetings, they got inspired and got into contact with another federated team to know how they built a feature. Therefore, this requirement has been met. Moreover, it was found that the developers of the federated team preferred to join well-prepared and interesting presentations, which was the case when the researcher prepared and promoted the presentations. However, after the researcher left the case company, meetings were still organized, but the preparation quality was lower, resulting in reduced motivation and attendance of the federated teams. As a result, it is necessary to do manual work as a central team for preparing the meetings and maintaining the community. Therefore this requirement is partially fulfilled.

The security self-assessment (D) was designed to improve the security part of the artefact requirements. The security officer mentioned that she still wanted to be involved in the first and more extensive deployments of applications. Still, the artefact helped to reduce the work in the first place because the guidelines were explained during the first assessment. Moreover, since there were hyperlinks with explanations that explain how to implement these security measures, security is improved. These hyperlinks are also linked to the pages which explain the development process. In this way, a holistic set of artefacts was produced that were linked to each other.

### 6.2.3 Effects in the context

The effects in the context are strongly related to the artefact requirements and the evaluation criteria of Hevner et al. (2004). Therefore, each mechanism artefact is elaborated according to the effects and these criteria. To support the evaluations, quotes from the focus groups are shown in *Table 19*.

First, according to the participants, the quality of the explanation and transparency of the development process was sufficient since all information could be found in one place. Furthermore, the explanation provided clear stages and detailed processes and the federated teams understood what to do without asking the central team since most information could be found. This resulted in an understanding of the processes and development by the federated teams that should be followed. Therefore, the efficacy of this mechanism was also covered. However, some points were still unclear, but they could be checked with the SPOC. Therefore, having a SPOC was also an important enabler in making sure that the process was followed. Moreover, when looking at the utility, new federated teams in the case company relied on third-

party vendors to bring in the experience, as shown in *Figure 18*. However, since these vendors have their own ways of working and they often do not speak proper English, communication with them is more complicated as well as transferring the development guardrails. This barrier is also similar to the identified communication barrier of Tufekci (2010) in GSD.

Secondly, when evaluating the quality of the intake process, the respondents mentioned that there were too many questions in the assessment. Therefore, the barrier to submit this intake was high, while it was not always necessary to understand an application to this depth. This resulted in a situation where requestors, as well as central team members, sometimes wanted to bypass the intake procedure since it was more efficient to have a meeting. However, the quality of the developed intake process, in terms of the tool, was working well; the process used was clear and the dashboard appeared to be helpful. Regarding the efficacy, the participants mentioned that the intake process was helpful since it increased the demand of the intelligent automation team. Nevertheless, the submitted intakes predominantly came from other parts of the business that only wanted to solve problems within their business. Therefore, the efficacy of the artefact turned out to be primarily suitable for increasing demand for central development and to increase monitoring capabilities to track demand and progress. Moreover, the increased demand from the business also resulted in positive commotion and promotion of the intelligent automation team and its platforms across the enterprise. In this way, the federated program was also promoted. When looking at the utility of the artefact, the participants mentioned that they did not want to act as 'a police officer' to ask the federated teams to fill in an intake, as indicated by the quotes presented in *Table 19*. Moreover, there was no incentive for federated teams to fill in this intake. This also illuminates the risk that an intake would harm the design freedom of the federated teams, as discussed in *chapter 4*. Additionally, the participants mentioned that the best way to understand the federated team's project is still to have a code review or sessions with the SPOC that is aligned with the team.

Thirdly, the internal community was implemented to align federated teams and to support reusability by providing regular internal community meetings. When looking at the efficacy of the artefact, the participants mentioned that the presentations on specific presented topics were helpful for them. Therefore, the efficacy of the community meetings was sufficient, as was discussed in *subsection 6.2.2*. However, the quality of work that is put into the internal community did influence its outcome. Furthermore, the federated teams mentioned that when there were not a considerable number of developers joining the meeting, they would not like to participate. In parallel, the central team did not have sufficient time to prepare the meetings similar to the ones that the researcher prepared and, because of the low number of developers joining, there was less motivation to put more effort into preparing the meetings. Therefore, the quality and the number of people joining the meetings will influence the utility of the artefact. This was confirmed by participant 2, the no-code product owner, who also implemented a community including a higher number of developers, who mentioned that a community was working well and that employees' questions were answered by other internal employees from other federated teams. Thus, to reap the benefits of having a community, efforts should be placed in well-prepared meetings and maintaining a community. Furthermore, according to the participants, the shared community channels in Microsoft Teams were not used since it is still easier to contact the SPOC from the central team. Similar to the community meetings, participant 2 mentioned that this would work when the number of teams increases. This would result in a situation where the federated teams help each other in the online internal community.

Lastly, the security self-assessment was implemented to support security in a federated low-code governance model. Security and compliance risks have already been identified risks in low-code development as well as its relation to federated governance, as discussed in *subsection 3.2.1* and *4.2.2*. According to participant 7 (the central security officer), the security self-assessment was necessary to conduct, and this artefact made it easier to evaluate the federated teams. Therefore, the efficacy of this mechanism artefact was covered. The security self-assessment presented the results clearly to the

federated teams and the measures contained links to intranet pages and recorded community meeting videos to explain how to implement security components. Therefore, the mechanism was perceived as a tool that worked properly with a good level of quality. Moreover, the developers of the federated teams mentioned that it was clear to them what security measures they had to implement. Furthermore, the artefact was helpful since it allowed to have central storage where all security related issues from the federated teams were documented. This was necessary and useful for audit purposes to look back at what decisions were taken. This indicated that the tool yield utility. Next to these effects, there was a suggestion for improvement. According to the security officer, the questions in the assessments should differ for the first assessment and the second assessment before going into production. Since the design is not fully clear at the beginning of a project and may differ after the actual development, the in-depth assessment should occur at the pre-deployment stage. Therefore, the first part of the assessment will about educating the federated team on what should be taken into account. In contrast, the assessment before deployment should be more formal and strict. Therefore, this mechanism should act as an educational tool for the federated teams to reduce the work for the central security officer in the earlier stage of the development process.

**Table 19:** Quotes during confirmatory focus groups for the evaluation of the artefacts





Quotes per mechanism artefact		Group	
		A	B
<b>(A) Transparency and process explanation</b>			
<i>'I use it when onboarding a federated team to explain to them how the development works.'</i>		X	
<i>'I've sent the manual to a lot of developers lately. There the hyperlinks redirect the readers to explanations on the process, which helps.'</i>		X	
<i>So when I, for example, wanted to deploy, I went to Sharepoint and read the information what I had to do, so when I want to deploy, I will have a look at how I have to deploy, and I will discuss it with [the SPOC]'</i>			X
<i>'To be honest, I have never used it in a way that some people do like read it, and then apply it. I am more like learning by doing, and when I get stuck, I look it up, so I don't see it as a guideline.'</i>			X
<i>'I am more like, I apply it, but I will not learn it.'</i>			X
<i>'All the information was well documented on the Sharepoint, so it was clear what I have to do and what tickets I have to send. Sometimes, it was unclear what I have to put inside those things, but then I asked [the SPOC], and he provided me with the information, and then in 2 minutes it was done.'</i>			X
<b>(B) Intake process</b>			
<i>'Intake process questions needs to be stipulated more concise and clear.'</i>			X
<i>'The intakes together with the dashboard allows us to track and monitor demand in an easier way which is really beneficial.'</i>			X
<i>'Business just thinks like "well, we do not have sufficient capability and time at the moment, let's put it at the intelligent automation team and let them solve it"'</i>			X
<i>'If there is no added value for the federated team, they will not fill in this assessment.'</i>			X
<i>'Yes, and I think that this assessment gives us an idea of what should be developed. However, when we do want to know more about the developments of a federated team, we will still plan in a meeting.'</i>			X
<b>(C) Community</b>			
<i>'A community helps, but currently, there are not sufficient teams doing OutSystems'</i>			X
<i>'Because there were not many people, sometimes I had the feeling that they were there because of us while we organize it for them.'</i>			X
<i>'During that meeting, I saw a feature, and then I thought; "ah wow, this is interesting, I need to make the same on my application", so I wanted to contact him.'</i>			X
<i>'Like a thirty to forty meeting with explanations on an integration where I can ask technical questions really helps me because I can spend hours on the internet to find it but it would be more sufficient if I can have this thirty-minute meeting with a colleague where I can ask all my questions'</i>			X
<i>'It does not have to be 50 persons, but like 10 to 15 would make it already more interactive.'</i>			X
<b>(D) Security Assessment</b>			
<i>'I think this monitoring is necessary; we need to document the decisions we make with regards to security and to clarify on central level what is necessary to have in place.'</i>			X

<i>'But I think to understand the security risks properly, we need to have the full design available and know exactly how it is built.'</i>	X
<i>'I still think it is a good moment to have a discussion and share thoughts on what measures should be taken into account and what risks are considered.'</i>	X
<i>'Especially in the beginning of the federated teams, it is useful, however after a couple of years, I think it is not useful for me anymore since I already know what to do regarding security when having, for example, a high-risk level on something.'</i>	X
<i>'The template and the process is there to trigger if there are more discussions are needed.'</i>	X
<i>'I think the template in itself has a lot of value since we, at least, have documented something that we have in our audit trail in case there are questions. And the discussions itself, and that is why I think there always should be a security officer involved if it is a more complicated app, that is the most valuable if you are properly securing your app.'</i>	X
<i>'if you have several options, concerns or doubts about handling security, then the discussion is much more important'</i>	X

### 6.3 Evaluation conclusions

The artefacts were created to support the governance of federated low-code development teams. In addition, as shown in *Table 16*, the artefacts should support the predefined artefact requirements. An overview of the results of the findings can be found in *Table 20*.

**Table 20:** Evaluation findings - effects of mechanism artefacts in the context of the case company

Mechanism artefacts			
 (A) Explanation and transparency of the development process	 (B) An intake process	 (C) An internal community	 (D) Security self-assessment
Strengths			
<ul style="list-style-type: none"> <li>• Improves explanation of development process</li> <li>• Reduces work for the central team</li> </ul>	<ul style="list-style-type: none"> <li>• Improves monitoring of demand &amp; progress</li> <li>• Increases central demand</li> <li>• Improves federated program commotion</li> </ul>	<ul style="list-style-type: none"> <li>• Inspires, aligns and supports reusability among federated teams</li> </ul>	<ul style="list-style-type: none"> <li>• Improves security monitoring of applications</li> <li>• Educates security guidelines before the development process</li> </ul>
Weaknesses			
<ul style="list-style-type: none"> <li>• Cannot ensure quality</li> </ul>	<ul style="list-style-type: none"> <li>• Not useful for understanding complete project</li> </ul>	<ul style="list-style-type: none"> <li>• Not effective with a low number of federated teams</li> <li>• Meetings should be prepared</li> </ul>	<ul style="list-style-type: none"> <li>• Should still be used complementary to the security process</li> </ul>

The findings of the focus groups showed that the artefact requirements were met in which each mechanism artefact had its share. Additionally, for each mechanism utility, quality and efficacy were validated. The mechanism artefacts had various effects as shown in *Table 16*. The explanation and transparency of the development process were used and appeared to be useful to explain how the development process worked. However, this is still an explanation and cannot ensure quality by implementing this. Therefore, the central team should emphasize using this process. Next, the intake process appeared to be useful for processing and understanding demand from business and to promote the team across the enterprise. However, it was not useful for understanding the project and there was no clear incentive for the federated team to submit an intake. Moreover, another goal for the intake process was to identify common components for applications that have to be developed, however, it appeared that more in-depth knowledge is needed for this. Therefore, a meeting or code review was still more sufficient to use since there are too many dependencies to review if a common component can be reused which cannot be covered in an intake. However, an intake could be the first line of identifying if components can be used. Community meetings appeared to be more helpful for inspiring federated teams to reuse components. Likewise, well-prepared community meetings were useful to inspire, align and



support reusability among federated teams. Moreover, when the number of teams within this internal community would increase, positive effects would increase too. Lastly, security-self assessments in the context of low-code development allowed federated teams to evaluate their developed product and understand what security measures have to be implemented. This tool was effective but should still be used complementary to regular manual security assessments when having larger and more risky applications.



# 7. Discussion and Conclusion

In this chapter, the results of the sub-research questions are discussed together with a conclusion on the main research question in *section 7.1*. Moreover, in this section, it is discussed how the artefact achieved the stakeholder goals, as defined in *Table 3*. Thereafter, the theoretical implications are discussed in *section 7.2*, and the practical implications are presented in *section 7.3*. Subsequently, the limitations of the research are debated in *section 7.4*. Lastly, recommended future research is discussed in *section 7.5*.

## 7.1 Discussion and conclusion

As discussed in *section 1.3*, the problem statement was as follows: *'it remains unclear how enterprises should govern federated development teams under a low-code paradigm. Currently, the right collaboration between the central teams and these federated teams is not yet defined resulting in a risky, unautomated, and unscalable situation where federated teams do not know what to do'*. Therefore, the objective was to improve the collaboration between federated teams and the central team by setting up a governance model such that the federated low-code teams understand what to do in order to establish a more secure and reusable development process in which repeated manual work could be reduced. Therefore the following main question was formulated:

**MQ:** *'How should enterprises govern federated development teams under a low-code paradigm?'*

The design science research methodology by Peffers et al. (2007) is used to answer this. The study resulted in the design of a set of artefacts which were a proposed development process, operating model and mechanism artefacts. A broad explorative case study with external interviewees from various disciplines was conducted to analyze how the operating model and processes should look and what mechanisms could be used to support the governance model. The findings of each sub-research question are presented and discussed below.

**SRQ1:** *What characterises the low-code paradigm?*

Based on a set of relevant scientific papers, ten characteristics were found that define the low-code paradigm. These characteristics are (i) *Ease of use*, (ii) *Types of users*, (iii) *Reusability*, (iv) *Development method*, (v) *Easy management and monitoring*, (vi) *Collaboration*, (vii) *Interoperability with other systems*, (viii) *Scalability*, (ix) *Security*, and (x) *Speed of development*. These characteristics of low-code development define in which way this new development method brings value to software development and in which way it is different from traditional methods. Especially, the new type of user, the development method, and the ease of development were important in the low-code paradigm. Apart from these characteristics, limitations and risks were found that are involved in the low-code paradigm; (i) *Type of users*, (ii) *Extensibility*, (iii) *Interoperability among LCAP*, and (iv) *Compliance and security*. The type of users may have less experience than full-stack developers, which might lead to worse designs. Furthermore, the extensibility is a limitation; although LCAP offer many design options, there is a specific limit for development. Additionally, there are still risks to security and compliance within low-code developments. The output of this sub-research question provided the researcher with a knowledge base of the technology to adjust the design of the governance model specifically to this new way of software development.

**SRQ2:** *What characterises federated governance, and how can federated software development be governed?*

First, it was analysed how federated structures work in general and why federated structures are beneficial. It was studied how governance can be set up and what best practices can be used based on

existing IT governance literature. Therefore, the framework and guidelines of Weill & Ross (2004a) were used to formulate IT governance. According to the authors; *'enterprises implement their governance arrangements through a set of governance mechanisms, structures, processes, and communications. Well designed, well-understood, and transparent mechanisms promote desirable IT behaviours'* (Weill & Ross, 2004a, p. 85). Furthermore, since governance should be defined on each level, a comparison was made with a centre of excellence since this is a commonly used term in practice when governing software developments in a federated way. Therefore, it was researched what this desired behaviour was on the platform level in a federated structure. To scope the research, it was looked at what tasks a central team has in a federated governance model since, in low-code, implementation often starts by having one central team. The literature research came down to eight categories of tasks that a central team has to perform in a federated software development model. These eight categories were later used as a starting point when categorizing the tasks and mechanisms in the context of specifically low-code development for sub-research question 4 (SRQ4). Since these tasks were later adjusted, they are not presented here. The output of this sub-research question formed a knowledge base regarding federated structures, and IT governance that was used as to develop a governance model.

**SRQ3:** *How does the low-code paradigm relate to federated development?*

The relationship between the low-code paradigm and a federated governance model was researched by interviewing persons who had an affinity with the low-code paradigm. The results of these interviews showed various benefits and risks. The discovered benefits were (i) *transparency in applications*, (ii) *more local developers*, (iii) *monitoring and access capabilities*, (iv) *quality assurance* and (v) *quick local development*. It was found that DevOps teams should be used in the context of a federated low-code governance model. This was due to the characteristics as defined in the first sub-research question. This implied that there should be no central operations team in these type of governance models and that the federated teams should autonomously take care of their applications. The found risks were related to (i) *compliance and security*, (ii) *lack of adoption of a new working method*, and (iii) *a lack of expertise*. Although DevOps teams may increase the quality of applications and mature federated teams, there were also still risks to design quality that should be mitigated by the central team. The found relations were used to define what the structures and processes would look like that should promote the desired behaviour.

**SRQ4:** *What mechanisms, tasks and best practices can be used to govern federated development teams under a low-code paradigm?*

Based on an explorative case study with a broad set of interviewees tasks and best practices were identified to explore how a federated model could be used to govern low-code teams. From one of the first interviews, it became clear that a federated governance model is even advised for low-code development:

*'At first, I would turn it around; I don't think low-code will be successful without a federated model'* – Respondent H, Chief Technology Officer low-code vendor

During the interviews, the identified tasks from SRQ2 were used to find what tasks were applicable in the context of low-code and what other tasks proved to be important in this context. The research resulted in a set of eight tasks that have to be executed by a central team in a federated low-code governance model. These tasks were: (i) *Defining a strategy, operating model & responsibilities*, (ii) *Communicating and aligning*, (iii) *Providing development guardrails and organizational policy*, (iv) *Knowledge sharing and educating*, (v) *Maintaining and managing reusable assets*, (vi) *Facilitating and maintaining technologies*, (vii) *Monitoring, assessing and evaluating*, and (viii) *Assisting and advising*. The latter was a category that was added and other tasks were renamed. Only one category, *Manage operations of deployed assets*, was removed. This was due to the fact that in a federated low-code context, DevOps should be used. These tasks remain very trivial and general as effective governance can still vary from company to company (Weill & Ross, 2004a). Furthermore, from these interviews, it appeared that how the governance of these federated teams looks, also depends on a set of factors. These factors are the (i) *Type of technology*, (ii)

*Type of organization*, (iii) *Type of application*, and (iv) *Maturity of the team*. A maturity and complexity model could help to categorize projects to teams with a corresponding maturity level. In this way, a central team can adjust the level of control it needs to have over the federated teams, based on the maturity level and the type of application. In this way, it is emphasized to have design freedom and autonomy within the federated teams. Therefore, central teams should have monitoring and evaluating capabilities to track the progress and experience level of federated teams.

As was found in **SRQ2**, to develop an effective governance model for low-code federated teams, mechanisms, processes, structures, and communication should be defined to ensure that the desired behaviour is conformed. During the explorative case study, interviews with consultants, low-code vendors and a cross-case analysis with other industries were used to discover best practices and ways to govern federated low-code teams. This information was used to design an operating model (structure), as shown in *Figure 17*, and the development process, as presented in *Figure 18*. Furthermore, mechanisms have been identified that support these two elements and nudge the federated teams to the desired behaviour. Moreover, these mechanisms could be implemented and evaluated easier in a case company. Therefore, additionally, four mechanism artefacts were developed that support this governance model, namely: An interactive explanation and transparency of the development process, an intake process, an internal community, and a security self-assessment. Because this study demonstrated and evaluated the artefacts in a case company, the artefacts were adjusted, which is explained in *chapter 6*.

Since the empirical study builds on top of the literature research, the proposed operating model and process resulted from combined studies. For the demonstration, an adjusted version of the proposed process (*Figure 19*) with four mechanism artefacts were demonstrated in a case environment and evaluated using two confirmatory focus groups. Moreover, a development process for starting teams was designed since this is the first level of a development process, and this type of team was available in the company to evaluate it with. In the end, the combined designed mechanism artefacts contributed in various ways to the governance of low-code development teams and were covering the artefact requirements that were defined at first instance. Furthermore, the artefacts mitigate the risks that were found in **SRQ3**. Nevertheless, not all stakeholder goals were met.

The defined stakeholder goals from *section 1.3* were: (A) *find a method that guides federated teams to improve software developments within federated teams*, (B) *define a governance structure that improves the collaboration between federated teams and the central team to increase software development quality within these teams*, (C) *gain insight into how this federated structure supports the design for reusability among federated teams*, and (D) *federated teams should be guided in a way that they understand what to do at what stage*. However, even though a governance model has been designed with complementary mechanisms, it was difficult to determine whether the software developments in the federated teams have improved. Though, the explanation of the development process and the security self-assessment ensured that the federated teams better understood what to do and the internal community brought the teams closer together during the community meetings. Furthermore, the explorative case study gained insights into best practices and other operating models to identify how the federated governance model could support reusability. Therefore, it can be concluded that two stakeholder goals were met (C and D) and two were partially met (A and C).

In the end, the internal community educated federated teams while reducing future manual work for the central team and inspired the federated teams, thereby encouraging reusability. The process explanations appeared to be helpful to clarify the way of working and was complementary to explaining the development guidelines to federated teams. Moreover, the security assessment helped to educate federated teams in the first instance, which would reduce manual work and increase the quality of applications. It also helped to monitor the conducted assessments in one place. Although the intake mechanism did not contribute to the fulfilment of the requirements and stakeholder goals, it was still useful for other purposes. Especially for the central team, it improved monitoring capabilities and it could track

the value of the team and the federated teams. Its initial goal of understanding projects, early on advising, and supporting reusability was not achieved. Existing review and advice methods remained the most preferred and useful mechanisms. But regardless of the tool, it remained important that the federated teams will be monitored and evaluated based on the application type and the maturity level in order to adjust the level of control and oversight of the central team. In this way, the task of the central team moved from governing federated teams to the process of governing federated teams.

When applying the study results to the governance matrix by Weill & Ross (2004a), still, decisions are taken on various archetypes in a federated low-code governance model. More specifically, a central team in a federated low-code governance model should take care of platform-related tasks, e.g. the infrastructure and platform performance and the general software architecture. The IT principles should be decided in an IT duopoly to define the strategy of the federated model and its operating model and processes which depend on the company. To make sure that the federated teams are aligned and the benefits of a federated model are achieved, a similarly clear vision among the teams should be defined (Lindström et al., 2017; Williams & Karahanna, 2013). Therefore, other stakeholders should be involved in this decision as well. In contrast, the federated team should be responsible for the identification, prioritization, and thus, investments of their applications and needs. However, investments and security on a platform level should be still decided by the central team. Because of this, a central team should understand and have a view of what is being developed by the federated teams to adjust control and enhance design freedom. This governance is shown in *Figure 18* where the central team is depicted with a C and the federated team with an F.

Decision \ Arche-type	IT Principles	IT Architecture	IT Infrastructure Strategies	Business Application Needs	IT Investment
Business Monarchy					
IT Monarchy		C	.....	C	
Feudal					
Federal				F	.....
IT Duopoly	C				
Anarchy					

*Figure 20:* Federated low-code governance implementation based on the IT Governance matrix by Weill & Ross (2004a)

## 7.2 Theoretical implications

This research contributed to the literature in several ways. Since low-code is a relatively new concept, not yet many studies written on this topic. Previous literature often focussed on the implications of low-code in industries or compared low-code vendors (Richardson et al., 2014; Sattar, 2018; Waszkowski, 2019). This research contributed to the scientific literature by providing a holistic view of the characteristics of the low-code paradigm specifically for leading LCAP. Secondly, the research provided a summary of generic tasks that should be executed by a central team when governing federated low-code teams. Thirdly, based on an explorative case study with various interviewees, a list is presented with relations between federated governance and low-code, which had not been investigated yet. Fourthly, based on a set of interviewees, the research provided insights into the dependencies of low-code governance models. These dependencies can be used in the literature to further investigate the influence of the dependencies on governance models and corresponding development processes. Fifthly, this research provided insights into the effects of the developed mechanisms in the context of federated low-code governance. Lastly, the

study contributed to both literature and practice by providing a holistic approach including a development process and operating model to effectively govern low-code federated development teams. Although, IT governance was widely researched (Stephen J. Andriole, 2012; Weill & Ross, 2005, 2004a), it was not yet investigated how it could be applied in the context of this new low-code paradigm.

### 7.3 Practical implications

On top of these theoretical implications, the output of this research provided an operating model and a corresponding process with mechanisms to achieve the desired behaviour, which is based on an explorative study, the output guides organizations that want to implement a similar model. Therefore, this generic governance model and the set of mechanism artefacts could be demonstrated and validated in other case studies. In addition, since the study was conducted with a broad set of interviews, factors that were included in this model were elements that could be generalized. Therefore, this study may serve as a starting point for organizations to build out their low-code capabilities. Additionally, the study provided a list of dependencies that influence the level of oversight of a central team. Therefore, this information can be used by other companies to create a maturity and complexity model with corresponding development processes to ensure design freedom within their federated teams. Furthermore, the artefact evaluation provides proof for other organizations to implement similar mechanisms to improve their federated low-code governance model. As an example, the security self-assessment did increase security awareness and educated federated teams and it is a tool that can be implemented in the context of low-code development or other software development models. The other effects were discussed in *section 6.2* and *Table 20*. Lastly, the study also presented insights into operating models of other organizations as well as a list of mechanisms that can be developed in a similar context.

### 7.4 Limitations

Similar to all studies, there are limitations in this research. For this study, a distinction is made between two types of limitations of this research. First, the research process contains limitations, and second, the proposed development process, operating model, mechanism artefacts, and findings have limitations.

#### 7.4.1 Research limitations

First, this study's scope focused on the processes, operating model, and mechanisms to achieve the desired behaviour. However, other elements of governance, e.g. cultures within enterprises may also influence the governance model. The time constraint urged to limit the scope of this thesis to demonstrate a subset of mechanisms and structures: the intake process, the explanation and transparency of the development process, the security self-assessment, and the internal community, while these are a subset of all mechanisms that promote the desired behaviour. In addition, there could be several ways to design these mechanisms and each design could lead to slightly different outcomes, e.g. other questions could be formulated during the intake process or other features could be added to the security self-assessment. However, during the development of the mechanism artefacts, it was iteratively designed with feedback loops with stakeholders from the case company to adjust it to this context to design it as helpful as possible. In addition, there could be more mechanisms identified during the explorative case study if other or more interviewees were selected. However, since a broad set of interviewees is selected, the list of mechanisms can be seen as holistic and generalizable.

Next, the artefacts were demonstrated at only one case company where the researcher also derived the initial problem statement for the start of this research. During the development of this study, the researcher may be biased because the researcher could have become too close to the case company (Shull et al., 2008). In this way, the researcher could have become dependent on the experiences within the case company which could have influenced the research. However, to incorporate reliability, other sources were interviewed, and interviewees with a lot of experience were selected. Furthermore, due to the current situation with the pandemic (see *Appendix V*), the interviews were conducted via video-calling instead of in-person meetings, which could cause a bias (Knox & Burkard, 2009). Because of these two possible biases,

there is a risk that the researcher may have influenced the interviewees (Shull et al., 2008). Moreover, the artefact evaluations only suggested that they work in the environment of the case company. However, the designed artefacts are not yet validated for other industries as well. However, because the artefacts are based on a broad set of interviewees, external validity is embedded into this research.

Moreover, due to time constraints for this research, it was not possible to review each interviewee's complete structure and operating model. Therefore the output of these models may not be entirely correct. However, the visualized models and descriptions were sent back to the interviewees for validation. Though, not all interviewees responded. Additionally, as explained, low-code is an emerging technology, and the low-code paradigm may shift to other characteristics over time. Therefore, over time, the output of **SRQ1** could become outdated and should be reviewed at a later point in time. Additionally, apart from *Definition 2*, the maturity level of the team that was used for the design of the mechanism artefacts was not clearly defined. In other words, the successful demonstration occurred with only one federated team with a specific maturity level (e.g. team composition and developers experience). Therefore, the implementation may not work in other contexts because of the undefined maturity level and difference in the team.

#### 7.4.2 Artefact limitations

The developed artefacts also had its limitations. After the evaluation, the quality of the artefacts was reviewed with two focus groups consisting of stakeholders that used the artefacts. From these evaluations, points of improvement were found. Therefore, after the improvement of the artefacts, other results may arise. Moreover, one of the advantages of having a federated model is to be able to reuse artefacts. However, it could also be possible that the federated teams do not want to share their applications or knowledge (Roth, 2014). It is crucial that the federated teams want to share their solutions in order to be reusable. If they keep the applications to themselves, the development could be increased locally, but the efficiency (time saved by other regions by developing a similar product) is not reached.

As discussed in the evaluation, the intake process was not specifically helpful to understand the projects of federated teams in the case company, but it served other purposes. Currently, a SPOC is a preferred way to assess the application to be developed. However, when the number of federated teams rises, this has to be scaled too, and the central capabilities cannot conduct these manual reviews of applications by SPOCS anymore. Therefore, it can be researched how this intake process works in a low-code context with the right incentives and what the results are if an intake is more flexible and adjustable to the federated teams. This also accounts for the effectiveness of a community with more federated teams. Furthermore, multiple artefacts were demonstrated and evaluated during this research that were also related to each other. Therefore, it could not be determined what the impact of one mechanism was. The security self-assessment helped to educate federated teams on the measures that should be implemented depending on the security level of their applications. However, there was still a risk that the federated team will wrongly fill in the assessment, and therefore, receive incorrect measures. Thus, they could be informed incorrectly. In order to solve this, the assessments were always sent to the security officer too. Therefore, the security officer could get into contact when it was unsure if the assessment was filled in correctly. Furthermore, for starting teams it will always be checked, and, for more mature teams, security self-assessments should be randomly monitored. Additionally, it is not yet determined in which cases federated teams should submit a new intake. This could be depending on e.g. the number of adjustments or features added, integrations, size, users added. This could be further researched as well as variations for intake processes on later levels.

#### 7.5 Future research

In this research, a subset of mechanisms has been designed and evaluated. However, future research is needed to validate if these mechanisms also provide similar outputs in other companies to have a reliable result. In this way, the robustness of the found results of this research would be improved. Furthermore, future research could focus on identifying if there are additional mechanisms that could support a low-

code federated governance model. In addition, as discussed in *subsection 4.3.2*, the governance model differs depending on a set of factors. Future research could identify which other factors could play a role in determining the maturity of a federated low-code team and to what levels an application can be categorized depending on, e.g. the complexity, type and amount of integrations and size. The proposed development process for governing federated low-code teams, as can be found in *Figure 17*, showed that the development phase and the central oversight differs depending on a set of factors, as presented in *subsection 4.3.2*. Since this study assessed the process focused on only one type of federated teams, future research could focus on processes that will be followed after getting to a new maturity level, after an X amount of development cycles, or when developing an application from category Y. Likewise, future research could investigate what maturity levels and complexity levels should exist in these governance models. Moreover, it can be researched how automated assessments linked to the maturity and complexity matrix can reduce the manual project assessments of the federated teams. Furthermore, the intake process appeared to be less valuable in a federated model where a SPOC is still feasible due to a lower number of federated teams. However, future research could focus on the impact of an intake process with a larger number of federated teams where the manual work issue is resolved with the intake and where the artefact is adjusted to the recommendations provided in *section 6.2*. Moreover, the effects of the intake process should be reviewed over a longer period to evaluate the effects on reusability and the impact of central architectural involvement.

Furthermore, because the explorative case study was conducted with interviewees from different disciplines and industries, a holistic understanding of the governance problems within low-code federated models has been obtained. Therefore, interesting topics for future research were found too. For example, when having multiple federated teams developing applications, a clear overview of components should be created. However, it should be assessed which components or applications could be valuable and beneficial for others and which ones could be left out. This assessment could be incorporated into the intake process to ensure that the applications that are being built will be designed with the right architectural foundations or components. By understanding the proposed application before the solution development phase, false positives and false negatives could be filtered out at an earlier stage to reduce time and costs. In addition, as explained in *chapter 4*, the federated program's value should be determined by having a clear view of the success and value. Therefore, KPI's to determine value should be established for low-code projects for each maturity level, taking into account the developers' learning curve. Examples could be entrepreneurial success, freedom of design, the number of automated processes, customer satisfaction, or satisfaction of federated developer colleagues. Another significant limitation of federated low-code governance is when an application becomes too big, risky, complex or when a federated team or its developers leave. In these cases application(s) have to be transferred to the central development team. When this happens, it should be clear and general policies around this have not yet been defined in the literature. It should be analyzed when there are too many risks and how this transfer can be managed in the best way. This could be combined with the maturity and complexity matrix as elaborated in *section 5.3*.

Furthermore, when the number of federated teams increases, the number of code reviews that the central team must perform to assess the quality of an application, also increases. Therefore, there might be insufficient capacity to conduct the code reviews at a certain point, and an automated code review and test framework could be implemented in a CI/CD pipeline. However, it is unclear at what stage enterprises have to implement this mechanism and at what point this mechanism should replace manual code reviews. Another option would be similar to the community reviews of case company A where mature developers can review and approve developed applications. This would increase the number of roles that can approve the deployment of applications which would speed up developments. Therefore, future research could focus on what requirements are needed, what the effects are in the context of low-code, what the risks are and how they should be implemented.



Another problem identified during the interviews was that currently there was no tool for local federated teams to identify new opportunities for developing applications. This tool could be developed by the central team and used by the federated teams. The tool could be linked to the existing applications to ensure reusability is supported and used as efficiently as possible. In addition, other researchers may focus on defining a value framework or building a tool to measure the success of a developed low-code application. This tool should be used within the last stage (the *Support & production stage*). In this way, a standardized value is measured among all applications, and reasonable monitoring of the federated model's value can be attained.



## 8. References

- Afsarmanesh, H., & Camarinha-Matos, L. M. (1997). Federated information management for cooperative virtual organizations. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1308(1308), 561–572. <https://doi.org/10.1007/bfb0022064>
- Aier, S., Gleichauf, B., & Winter, R. (2001). *Understanding Enterprise Architecture Management Design - An Empirical Analysis*. 645–654. <http://www.alexandria.unisg.ch/Publikationen/71084>.
- Alexander, F. (2019). *Application Platform as a Service: What Is It Really?* Outsystems Blog. <https://www.outsystems.com/blog/posts/application-platform-as-a-service/>
- Altintas, N. I., Cetin, S., & Dogru, A. H. (2007). Industrializing software development: The “factory automation” way. *International Conference on Trends in Enterprise Application Architecture*, 4473 LNCS, 54–68. [https://doi.org/10.1007/978-3-540-75912-6\\_5](https://doi.org/10.1007/978-3-540-75912-6_5)
- Anagnoste, S. (2013). Setting Up a Robotic Process Automation Center of Excellence. *Management Dynamics in the Knowledge Economy*, 6(2), 307–322. <https://doi.org/10.25019/mdke/6.2.07>
- Andriole, S. J. (2014). *Ready Technology: Fast-tracking New Business Technologies*. CRC Press.
- Andriole, Stephen J. (2012). Managing technology in a 2.0 world. *IT Professional*, 14(1), 50–57. <https://doi.org/10.1109/MITP.2012.13>
- Bernaschina, C., Comai, S., & Fraternali, P. (2018). The Journal of Systems and Software Formal semantics of OMG’s Interaction Flow Modeling Language (IFML) for mobile and rich-client application model driven development. *Journal of Systems and Software*, 137, 239–260.
- Biggins, J. (2018). *Designing the right operating model for your automation Center of Excellence*. Infosys Consulting. <https://www.infosysconsultinginsights.com/2018/01/18/automation-centers-of-excellence/>
- Bourgault, M., Drouin, N., & Hamel, É. (2008). Decision Making Within Distributed Project Teams: An Exploration of Formalization and Autonomy as Determinants of Success. *Project Management Journal*, 39, 97–110. <https://doi.org/10.1002/pmj>
- Brown, C. V. (1999). Horizontal Mechanisms under Differing IS Organization Contexts. *MIS Quarterly: Management Information Systems*, 23(3), 421–454. <https://doi.org/10.2307/249470>
- Cabot, J. (2020). Positioning of the low-code movement within the field of model-driven engineering. *Proceedings - 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS-C 2020 - Companion Proceedings*, 535–537. <https://doi.org/10.1145/3417990.3420210>
- Chen, D., Doumeingts, G., & Vernadat, F. (2008). Architectures for enterprise integration and interoperability: Past, present and future. *Computers in Industry*, 59(7), 647–659. <https://doi.org/10.1016/j.compind.2007.12.016>
- Cisco. (2020). *What is Shadow IT?* <https://www.cisco.com/c/en/us/products/security/what-is-shadow-it.html>
- Colantoni, A., Berardinelli, L., & Wimmer, M. (2020). DevOpsML: Towards Modeling DevOps Processes and Platforms. *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, 1–10.
- Dahm, J. P., Richards, D. F., Black, A., Bertsch, A. D., Grinberg, L., Karlin, I., Kokkila-Schumacher, S., Leon, E. A., Neely, J. R., Pankajakshan, R., & Pearce, O. (2019). Sierra Center of Excellence: Lessons learned. *IBM Journal of Research and Development*, 64(3–4), 1–15. <https://doi.org/10.1147/JRD.2019.2961069>
- De Vries, B. (2019). Low-code and the road to sustainable software. *Compact*. <https://www.compact.nl/articles/low-code-and-the-road-to-sustainable-software/>
- Di Rocco, J., Di Ruscio, D., Iovino, L., & Pierantonio, A. (2015). Collaborative repositories in model-driven engineering. *IEEE Software*, 32(3), 28–34. <https://doi.org/10.1109/MS.2015.61>
- DiCicco-Bloom, B., & Crabtree, B. F. (2006). The qualitative research interview. *Medical Education*, 40(4), 314–321. <https://doi.org/10.1111/j.1365-2929.2006.02418.x>
- Dresch, A., Lacerda, D. P., & Antunes, J. A. V. J. (2015). Design science research - A Method for Science and Technology Advancement. In *Computing Handbook, Third Edition: Information Systems and Information Technology*. Springer. <https://doi.org/10.1007/978-3-319-07374-3>
- Farwick, M., Agreiter, B., Breu, R., Ryll, S., Voges, K., & Hanschke, I. (2011). Automation processes for enterprise architecture management. *Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOC, June 2014*, 340–349. <https://doi.org/10.1109/EDOCW.2011.19>

- Faura, M. V. (2019). *The Many Flavors of “ Low-Code .”* InfoQ. <https://www.infoq.com/articles/many-flavors-low-code/>
- Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A. G., & Mehandjiev, N. (2004). Meta-design: A Manifesto For End-User Development. *Communications of the ACM*, 47(9), 33–37. <https://doi.org/10.1145/1015864.1015884>
- Fischer, R., Aier, S., & Winter, R. (2007). A federated approach to enterprise architecture model maintenance. *Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures - Concepts and Applications, EMISA 2007*, 2(2), 9–22.
- Frost, T. S., Birkinshaw, J. M., & Ensign, P. C. (2002). Centers of excellence in multinational corporations. *Strategic Management Journal*, 23(11), 997–1018. <https://doi.org/10.1002/smj.273>
- Furst, S. A., Reeves, M., Rosen, B., & Blackburn, R. S. (2004). Managing the life cycle of virtual teams. *Academy of Management Executive*, 18(2), 6–20. <https://doi.org/10.5465/AME.2004.13837468>
- Garita, C., Afsarmanesh, H., & Hertzberger, L. O. (2001). The PRODNET cooperative information management for industrial virtual enterprises. *Journal of Intelligent Manufacturing*, 12(2), 151–170. <https://doi.org/10.1023/A:1011252510739>
- Gartner. (2020). *Gartner Glossary Application Platform as a Service ( aPaaS )*. Information Technology. <https://www.gartner.com/en/information-technology/glossary/application-platform-as-a-service-apaas>
- Gibbons, M. (Ed. ). (1994). The new production of knowledge: The dynamics of science and research in contemporary societies. Sage. In *Great Britain: Sage Publications Ltd.*
- Gray, B. (2004). Informal Learning in an Online Community of Practice. *Journal of Distance Education*, 19(1), 20–35.
- Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly*, 28(1), 75–105. <https://doi.org/10.2307/25148625>
- Hoda, R. (2019). *Agile Processes in Software Engineering and Extreme Programming – Workshops*. <https://doi.org/10.1007/978-3-030-30126-2>
- ITGI. (2006). *Enterprise Value: Governance of IT Investments* (Issue May). Rolling Meadows.
- Jabbari, R., Ali, N., & Petersen, K. (2016). What is DevOps ? A Systematic Review on Definitions and Practices. *Proceedings of the Scientific Workshop Proceedings of XP2016 (XP ’16 Workshops)*.
- Khosroshahi, P. A., Aier, S., Haider, M., Roth, S., Matthes, F., & Winter, R. (2015). Success factors for federated enterprise architecture model management. *Lecture Notes in Business Information Processing*, 215, 413–425. [https://doi.org/10.1007/978-3-319-19243-7\\_38](https://doi.org/10.1007/978-3-319-19243-7_38)
- Kirschner, B., & Roth, S. (2014). Federated enterprise architecture model management: Collaborative model merging for repositories with loosely coupled schema and data. *Tagungsband Multikonferenz Wirtschaftsinformatik 2014, MKWI 2014*, 2163–2174.
- Knox, S., & Burkard, A. W. (2009). Qualitative research interviews. *Psychotherapy Research*, 19(4–5), 566–575. <https://doi.org/10.1080/10503300802702105>
- Kruit, R. (2018). *A Digital Transformation Framework: Start-Structure- Scale*. Mendix Blog. <https://www.mendix.com/blog/digital-transformation-framework/>
- Kumar, S., & Brouwer, A. (2020). Breaking the deadlock for low-code on the Dutch market. *Compact*, 55–63.
- Landy, F. J., & Conte, J. M. (2013). Work in the 21st century: An introduction to industrial and organizational psychology. In *Wiley & Sons* (4th ed., Vol. 369, Issue 1). <https://doi.org/10.1017/CBO9781107415324.004>
- Le Clair, C. (2017). *RPA Operating Models Should Be Light And Federated: A 10-Point Control Framework Helps Manage The Digital Workforce Of The Future*.
- Lefort, B., & Costa, V. (2019). *Benefits of Low Code Development Environments on Large Scale Control Systems*. 0–5.
- Lindström, N., Nyström, B., & Zdravkovic, J. (2017). An analysis of enterprise architecture for federated environments. *Lecture Notes in Business Information Processing*, 305, 156–170. [https://doi.org/10.1007/978-3-319-70241-4\\_11](https://doi.org/10.1007/978-3-319-70241-4_11)
- Linthicum, D. S. (2009). *Cloud computing and SOA convergence in your enterprise: a step-by-step guide*. Pearson Education.
- Marciniak, R. (2012). Center of Excellence as a next step for shared service center. *Journal of International Scientific Publication: Economy & Business, ISSN*, 1313–2555.
- Miles, M. B., & Huberman, Michael, A. (1994). Qualitative Data Analysis: A Methods Sourcebook: An Expanded Sourcebook. In *Sage Publishing, Thousand Oaks* (2nd ed.).
- Mintzberg, H. (1979). *The Structuring of Organizations: A Synthesis of the Research*. Prentice-Hall.
- Paasivaara, M., & Lassenius, C. (2004). The benefits and limitations of knowledge management in global software development. *3rd International Workshop on Global Software Development*, 42–47. <https://doi.org/10.1049/ic:20040316>

- Peppers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3), 45–77. <https://doi.org/10.2753/MIS0742-1222240302>
- Prikladnicki, R., & Yamaguti, M. H. (2004). Risk Management in Global Software Development : A Position Paper. *3rd International Workshop on Global Software Development*, 18–20.
- Richardson, C., Rymer, J. R., Mines, C., Cullen, A., & Whittaker, D. (2014, June). New Development Platforms Emerge For Customer-Facing Applications. *Forrester: Cambridge, MA, USA*, 15.
- Roth, S. (2014). *Federated Enterprise Architecture Model Management -- Conceptual Foundations, Collaborative Model Integration, and Software Support*. Technischen Universität München.
- Ruparelia, N. B. (2010). Software development lifecycle models. *ACM SIGSOFT Software Engineering Notes*, 35(3), 8–13. <https://doi.org/10.1145/1764810.1764814>
- Rymer, J. R., & Koplowitz, R. (2019). The Forrester Wave™ : Low-Code Development Platforms For AD&D Professionals, Q1 2019. In *Forrester: Cambridge, MA, USA, 2019*.
- Sahay, A., Indamutsa, A., Di Ruscio, D., & Pierantonio, A. (2020). Supporting the understanding and comparison of low-code development platforms. *Proceedings - 46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020*, 2(December), 171–178. <https://doi.org/10.1109/SEAA51224.2020.00036>
- Sambamurthy, V., & Zmud, R. W. (1999). Arrangements for information technology governance: A theory of multiple contingencies. *MIS Quarterly: Management Information Systems*, 23(2), 261–290. <https://doi.org/10.2307/249754>
- Sanchis, R., García-Perales, Ó., Fraile, F., & Poler, R. (2020). Low-code as enabler of digital transformation in manufacturing industry. *Applied Sciences (Switzerland)*, 10(1). <https://doi.org/10.3390/app10010012>
- Sattar, N. A. (2018). *Selection of Low-Code Platforms Based on Organization and Application Type*. Lappeenranta University of Technology.
- Saunders, M., Lewis, P., & Thornhill, A. (2012). *Research Methods for Business Students* (6th ed.).
- Seiner, R. S. (2017). Can You Federate Data Governance? *The Data Administration Newsletter*, 1–5. <https://tdan.com/can-you-federate-data-governance/21009>
- Sheth, A. P., & Larson, J. A. (1990). Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. In *ACM Computing Surveys (CSUR)* (Vol. 22, Issue 3, pp. 183–236). <https://doi.org/10.1016/j.jpeds.2011.05.052>
- Shull, F., Singer, J., & Sjøberg, D. I. (2008). *Guide to Advanced Empirical Software Engineering*. Springer Science & Business Media.
- Strassmann, P. A. (1995). Governance: The New IS Agenda. *Computerworld Leadership Series* (29:9). <https://www.strassmann.com/pubs/computerworld/governance.shtml>
- Tiemens, S., Kumar, S., & Brouwer, A. (2019, March). Low-code: empower the capability to accelerate. *Compact*, 31–37.
- Tiemens, S., & Weel, J. (2019). The low-code journey: from experimenting and discovering to delivering value at scale and continuous improvements. *KPMG*, 1–5. <https://home.kpmg/nl/nl/home/insights/2019/07/the-low-code-journey-from-experimenting-and-discovering-to-delivering-value-at-scale-and-continuous-improvements.html>
- Tisi, M., Mottu, J., Kolovos, D., De Lara, J., Guerra, E., Di Ruscio, D., Pierantonio, A., & Wimmer, M. (2019). *Lowcomote : Training the Next Generation of Experts in Scalable Low-Code Engineering Platforms*.
- Tjoa, A. M., Bellatreche, L., Biffi, S., Leeuwen, J. Van, Eds, J. W., & Steffen, B. (2018). *SOFSEM 2018 : Theory and Practice*.
- Tremblay, M. C., Hevner, A. R., & Berndt, D. J. (2010). Focus Groups for Artifact Refinement and Evaluation in Design Research. *Communications of the Association for Information Systems*, 26. <https://doi.org/10.17705/1cais.02627>
- Tsai, W. (2002). Social structure of “coopetition” within a multiunit organization: Coordination, competition, and intraorganizational knowledge sharing. *Organization Science*, 13(2), 179–190. <https://doi.org/10.1287/orsc.13.2.179.536>
- Tufekci, O., Cetin, S., & Arifoglu, A. (2010). Proposing a federated approach to global software development. *4th International Conference on Digital Society, ICDS 2010, Includes CYBERLAWS 2010: The 1st International Conference on Technical and Legal Aspects of the e-Society*, 150–157. <https://doi.org/10.1109/ICDS.2010.34>
- van Aken, J., Berends, H., & van der Bij, H. (2012). Problem Solving in Organizations. In *Problem Solving in Organizations* (Second Ed.). Cambridge University Press. <https://doi.org/10.1017/cbo9781139094351>
- van Brummelen, J., & Slenders, T. (2019, February). Modern Software Development: It is all about quality and

- speed. *Compact*, 1–15. <https://www.compact.nl/articles/modern-software-development/>
- Vincent, P., Iijima, K., Driver, M., Wong, J., & Natis, Y. (2020). Magic Quadrant for Enterprise Low-Code Application Platforms. *Gartner, September*, 1–33.
- Waszkowski, R. (2019). Low-code platform for automating business processes in manufacturing. *IFAC-PapersOnLine*, 52(10), 376–381. <https://doi.org/10.1016/j.ifacol.2019.10.060>
- Webster, J., & Watson, R. T. (2002). Analyzing the Past to Prepare for the Future: Writing a Literature Review. *MIS Quarterly*, 26(2), xiii–xxiii. <https://doi.org/10.1.1.104.6570>
- Weill, P., & Ross, J. (2005). A matrixed approach to designing IT governance. *MIT Sloan Management Review*, 46(2), 26–34.
- Weill, P., & Ross, J. W. (2004a). IT governance: How top performers manage IT decision rights for superior results. In *Harvard Business School Press*. <https://doi.org/10.1093/0195159535.003.0008>
- Weill, P., & Ross, J. W. (2004b). IT Governance on One Page. *MIT Sloan School of Management*, 349.
- Wenger, E. (1998). Communities of practice: Learning, meaning, and identity. In (*Learning in Doing: Social, Cognitive and Computational Perspectives*). Cambridge: Cambridge University Press. <https://doi.org/10.1017/CBO9780511803932>
- Williams, C. K., & Karahanna, E. (2013). Causal Explanation in the Coordinating Process: A Critical Realist Case Study of Federated IT Governance Structures. *MIS Quarterly*, 37(3), 933–964. <https://doi.org/10.1017/CBO9781107415324.004>
- Wu, Y., Wang, S., Bezemer, C. P., & Inoue, K. (2019). How do developers utilize source code from stack overflow? In *Empirical Software Engineering* (Vol. 24, Issue 2). Empirical Software Engineering. <https://doi.org/10.1007/s10664-018-9634-5>
- Yin, R. K. (2013). *Case study research: Design and methods* (5th ed.). Great Britain, USA: Sage Publications Inc.

Thank you,

Christophe Slangen

A stylized, handwritten signature in white ink, consisting of several overlapping loops and a long, sweeping tail that extends towards the right.



## 9. Appendixes

### **Appendix I**

The transcripts of the interviews are excluded from this research due to confidentiality reasons of the interviewees. The transcripts can be requested and can be shared after approval by the interviewee of the corresponding transcript.

## Appendix II

*-This figure is removed due to confidentiality reasons-*

**Figure 21:** Organization overview



### Appendix III

*-This figure is removed due to confidentiality reasons-*

**Figure 22:** Current state analysis of solution development process for case company before thesis

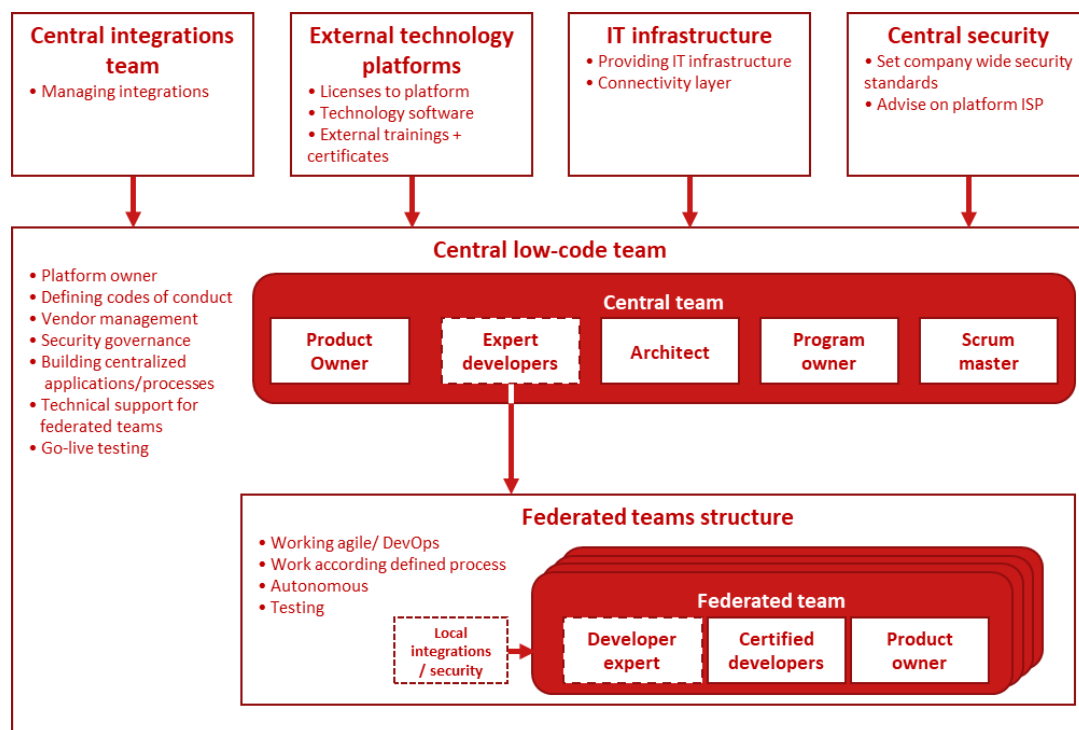


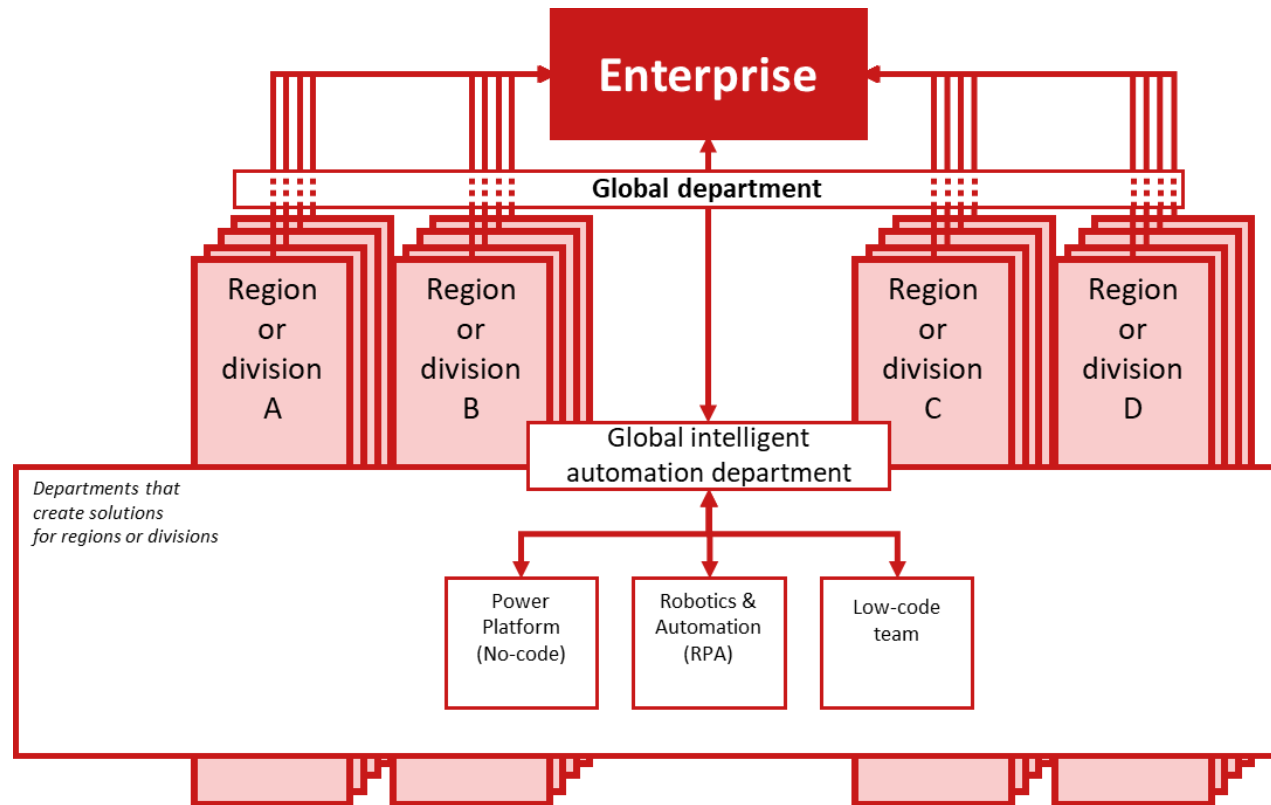
Figure 23: Operating model case company

**Notes:**

The low-code team is part of the central intelligent automation team including two other teams (RPA and No-code), as shown in *Figure 21*. As a construction to bring in expert knowledge, the case company used SPOC's that assisted and advised the federated teams. Other company-related information can be found in *section 1.2* and *section 5.2*.

### Centralized model

In the centralized structure, the three central teams that are part of the overarching global department are building applications on request for the regions or divisions. Therefore, this white rectangle covers all regions but is part of the global department.



## Appendix IV

**Table 21:** Artefact requirements for the case company

	Requirement	Description	Source
1	The artefact should support scalable situations	The artefact should be designed in a way that when more federated teams are established, no extra manual tasks are needed	Program owner
2	The artefact should be automated	The artefact should use as few operators from the central team as possible; manual activities should be mitigated	Program owner
3	The artefact should support for reusability	The artefact should not hinder the federated teams from reusing components or developed solutions	Product owner
4	The central team should not act as a strict authority	The artefact should be designed in a way that federated teams do not see the central team as a strict authority that controls everything	Product owner
5	The federated teams should be able to decide on the lifecycle of the solutions	This lifecycle is regarding the functional requirements of the application that is being developed	Product owner
6	The federated teams should be able to decide on the method used (e.g. agile/design sprints)	The federated team can decide in what way the application is built. However, they should obey specific policies of the central team	Product owner
7	Every solution created by a federated team should hand over an information classification	From a security perspective, each solution should bear in mind what the risks are for the solution that is being developed.	Security team
8	The charging of the federated model should be clear and visible	Since charging is an essential step in the federated model, this should be clear to the federated teams too	Program owner
9.	The central team should not 'hinder' the federated teams in the development process	The federated teams should have design freedom when developing their products.	Product owner

**Appendix V**

At the beginning of 2020, the world was hit by the pandemic of COVID-19. This pandemic caused many deaths, and because of the highly infectious characteristics of the virus, strict measures had been taken. These measures also influenced the way people work and how organizations operate. Because of the virus, in most countries, working from home became the new norm. Therefore need for digital transformation was even more necessary because almost all work had to be executed digitally. This pandemic had a tremendous impact on various industries (e.g. <https://www.volkskrant.nl/cs-bd2e8f07>).

Since this research was written during the pandemic, it also impacted this study. For example, all interviews, focus groups, document research and presentations were conducted virtually. Additionally, communication between all stakeholders was virtual too. This communication included the collaboration within the case company, the contact with the mentor, and artefact implementations.

## Appendix VI

### A Governance model – Banking

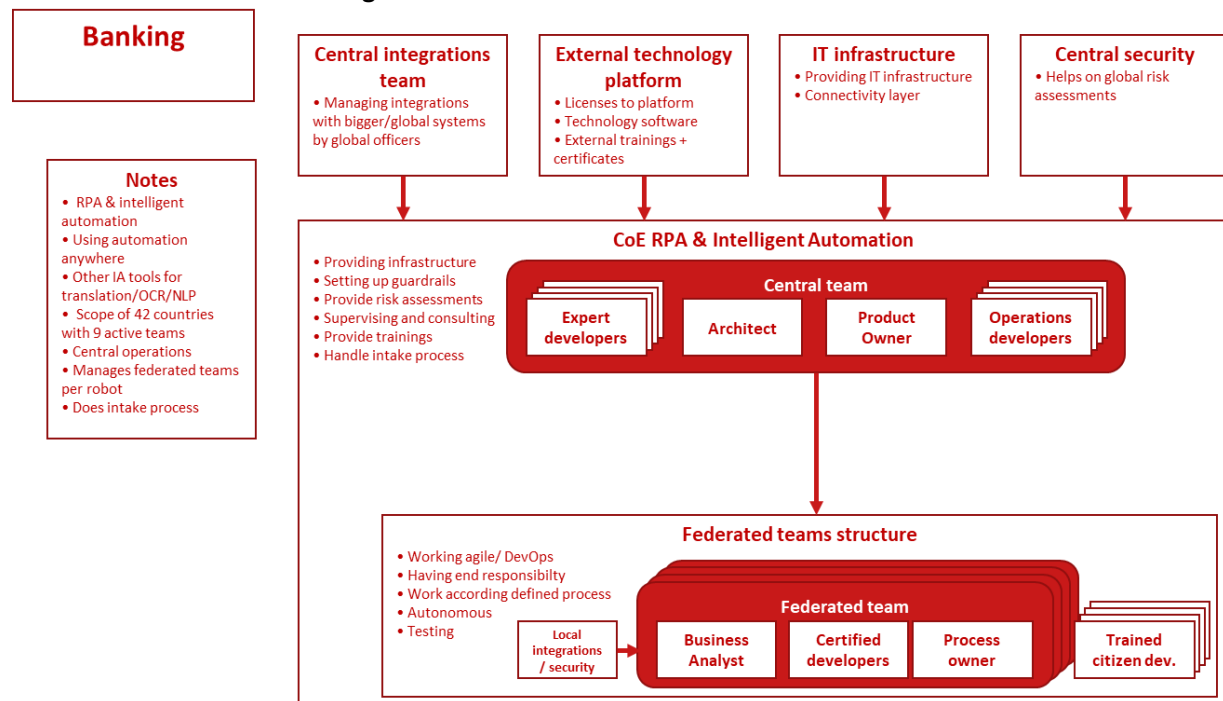


Figure 24: Governance model - Banking

#### Notes:

- Does code reviews/peer reviews among federated teams by mature developers
  - Validator role developer can do final code review before deployment
- A hybrid model with teams and citizen developers
- Training provided by CoE are technology and company methodology focussed
- The central team provides training:
  - Analyst training: 1.5-day course
  - Developer training: 3.5-day course
- A Federated team should consist of at least these two roles (because of seg. duties)
- During intake assessment, it covers parameters: suitability, feasibility, risk. *'The better the score, the more secure'*

## B. Governance model – Oil & Gas

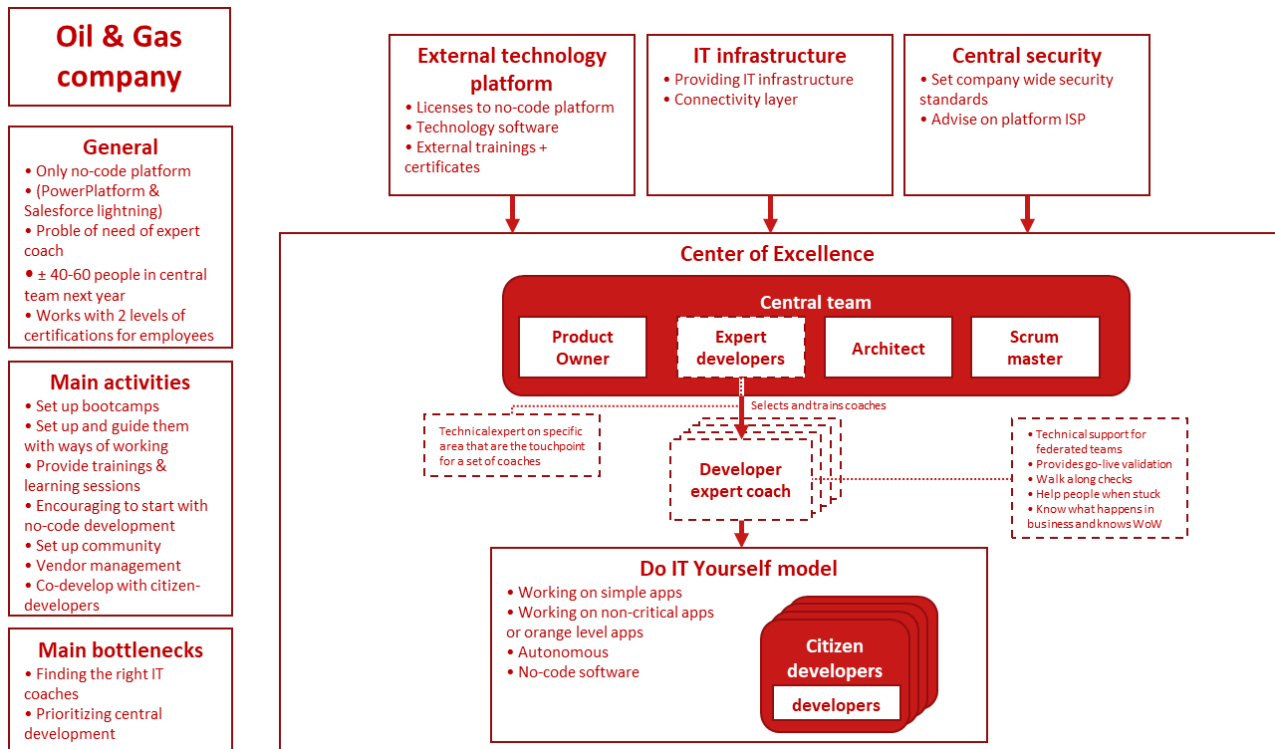


Figure 25: Governance model - Oil & Gas

### Notes:

- The enterprise used a zoning-model<sup>11</sup> to classify projects
  - Scope & Criticality VS Data & Complexity → see Figure 26
- Levels:
  - **Red:** Danger / off-limits. Business critical systems/ most confidential data / external facing.
    - It cannot be created by citizen dev.
  - **Yellow:** in between → co-developed with the central team
  - **Green:** a developer can develop everything they want within safe boundaries
- Coaches guide the citizen developers through this model

<sup>11</sup> Unfortunately the underlying assessment metrics could not be shared with the researcher.

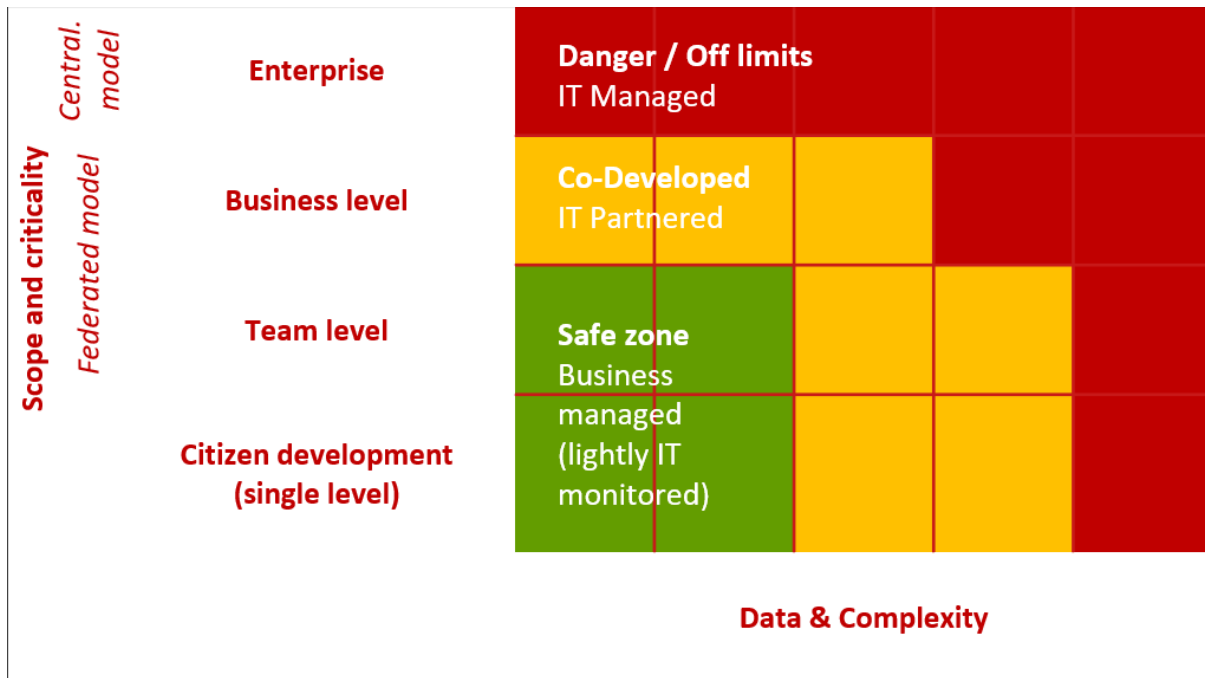


Figure 26: Zoning model by case company C

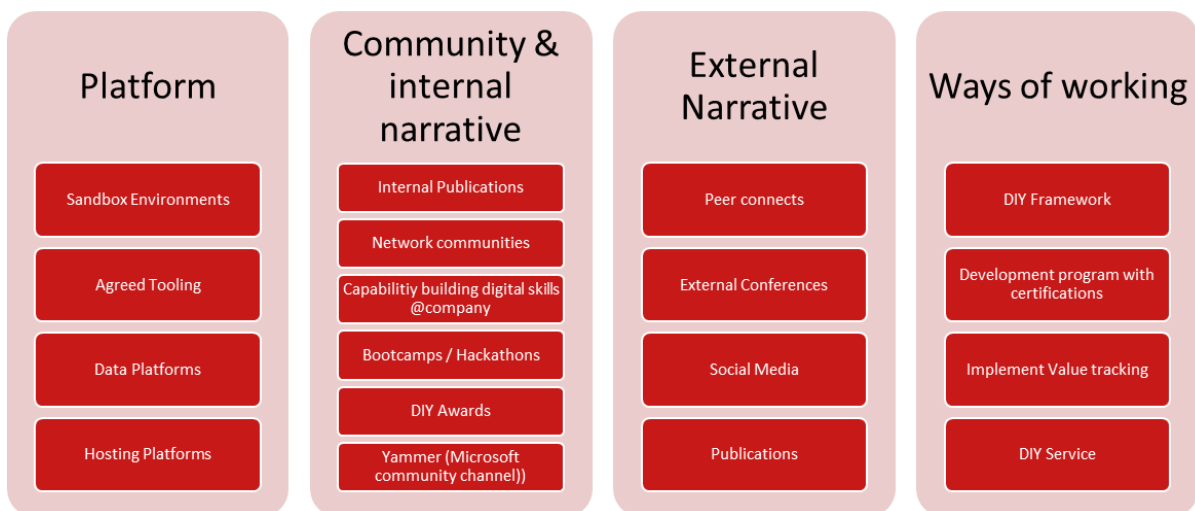


Figure 27: Main pillars central team case company C



### C. Governance model – Logistics & chemicals

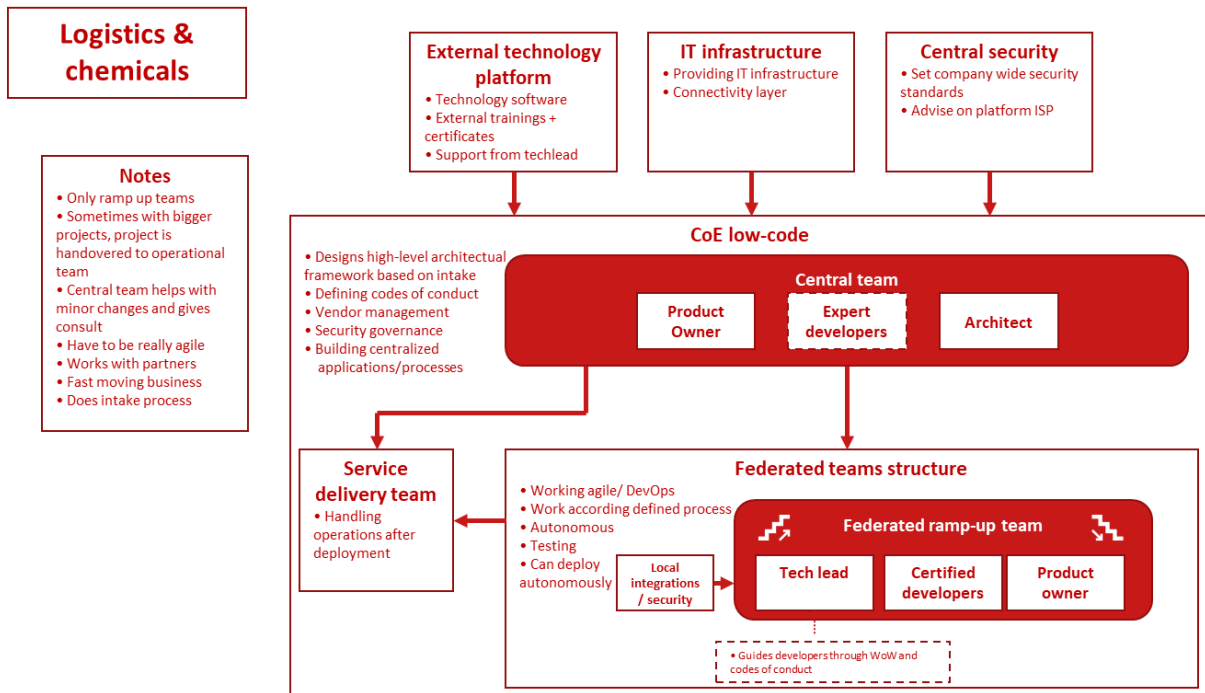


Figure 28: Governance model – Logistics & Chemicals

**Notes:**

- Used ramp-up teams that started from an intake assessment with a risk evaluation
- Smaller enterprise compared to the other companies (>5000 employees)

### D. Governance model – Banking 2

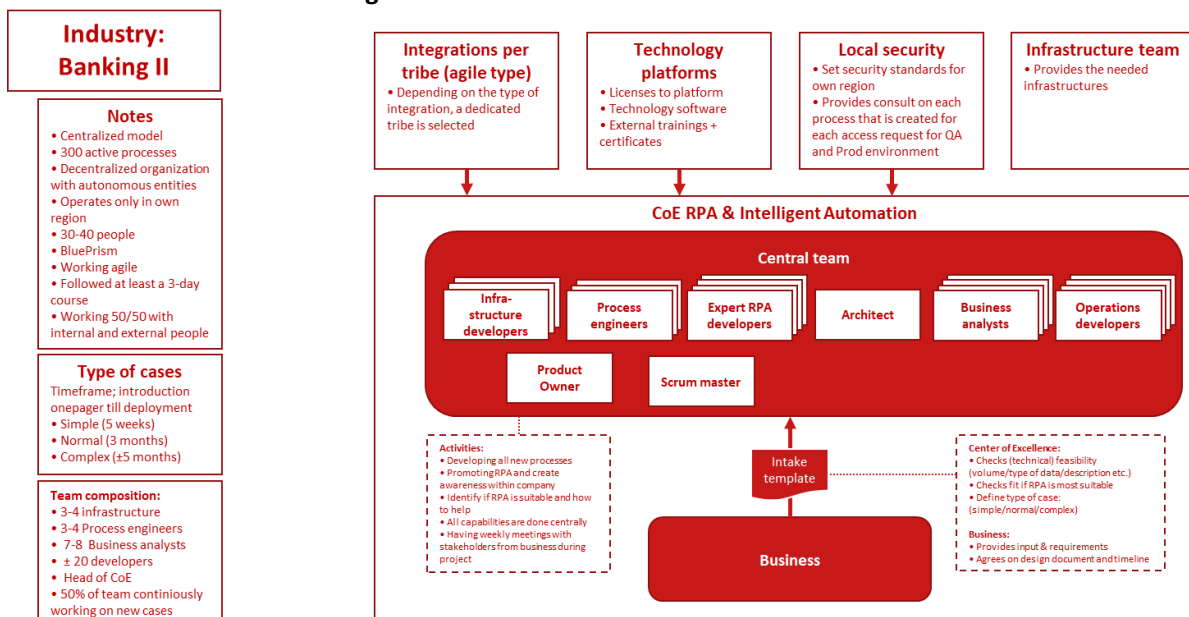


Figure 29: Governance model banking 2

## Appendix VII

According to an extensive study of governance structure among 256 firms, various models can be retrieved (Weill & Ross, 2004a). The ten most used governance model combinations were covering 25% of all firms. The three most successful governance model combinations can be found in *Figure 30*.

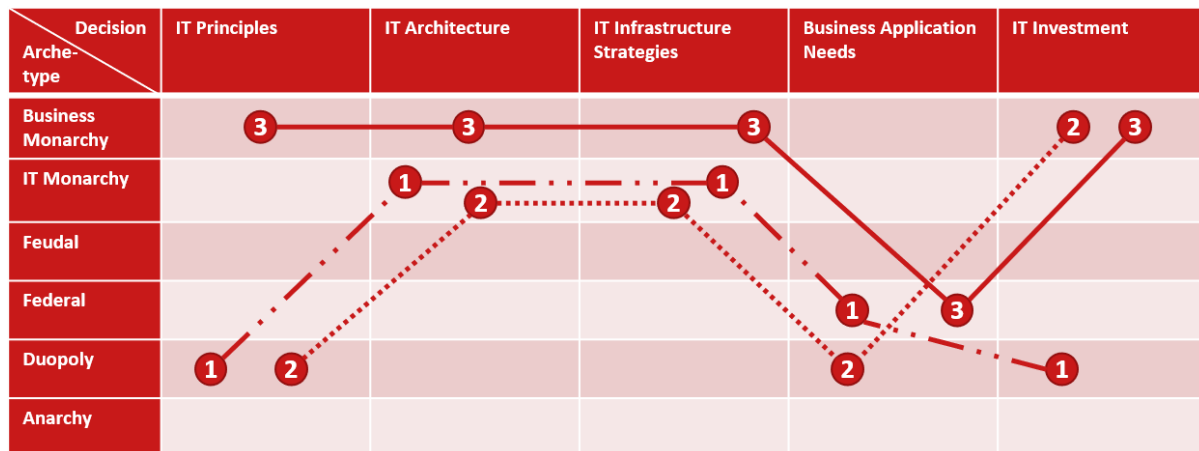


Figure 30: Top three governance performers by Weill & Ross (2004a)

### Appendix VIII

In this appendix, figures are presented that are not required in the main text but can support and explain the study.

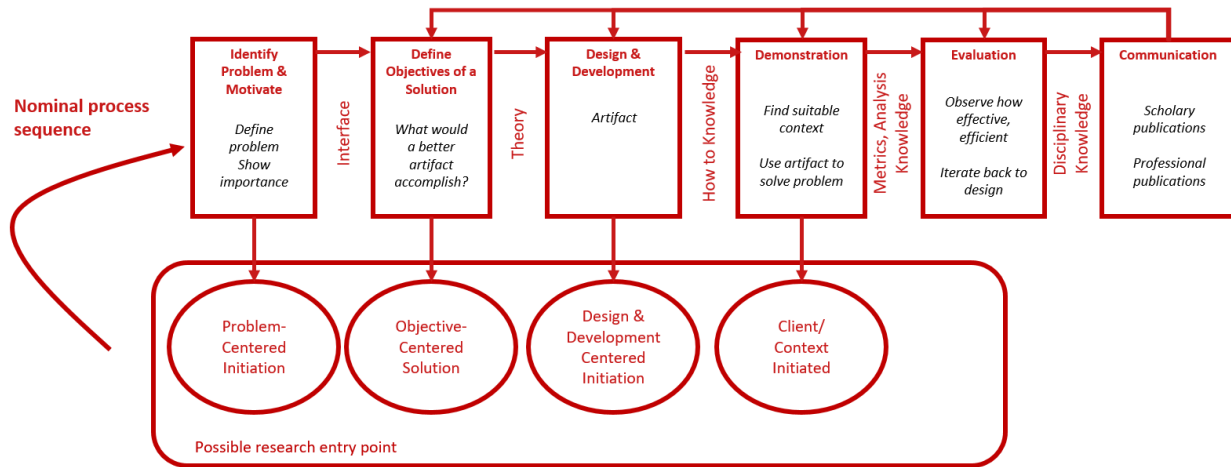


Figure 31: DSRM Process Model (Peffers et al., 2007)

## Appendix IIX

### A. Research protocol low-code

Below, the two interview protocols can be found that were used as a guide for the semi-structured interviews.

#### General introduction

Hi,

For my master thesis at the Eindhoven University of Technology, I am researching how enterprises can guide and govern federated development teams under a **low-code paradigm**. I would like to understand what aspects this low-code paradigm introduces to development teams in enterprises with this interview. I want to get insights into this vision from a low-code platform and expert perspective.

I kindly ask you to read the questions beforehand. To have a more concrete and compelling conversation, I would like to invite you to think about the aspects that low-code/no-code software development platforms introduce and how they should guide and govern these federated teams.

The interview will be conducted virtually via a Microsoft Teams meeting (preferably a video conversation) because of COVID-19 circumstances. All personal and company-related information is confidential, and the interview will be anonymized. A transcript of the conducted interviews will be sent after the interview to confirm the recorded data. The gathered information will be used as input for my master thesis for the Eindhoven University of Technology.

Thank you,

Christophe Slagen

#### Questions for the federated model

*[Explanation of the federated model to be aligned with the interviewee; the interviewer and interviewee should have a similar definition when talking about a subject before starting the interview.]*

##### General:

*This general information is needed to get an idea of the position and knowledge gathered from the interviewee.*

1. Could you tell me something about your background and your role in your company?

##### Low-code paradigm definition

*With these questions, the goal is to understand how the low-code platforms look at the low-code paradigm and get a clear view of what aspects are introduced. In what way would they describe this?*

2. What are the opportunities for low-code platforms in low-code federated development models?
3. How would you describe the low-code paradigm?

4. In what way do you think low-code platforms are beneficial when having federated development teams?
5. What are the challenges/limitations for low-code platforms in federated models?
6. What are, according to you, the most important aspects of low-code platforms that differentiate them from traditional coding?
7. Do you think new culture, practices and design approaches are also part of the shift to low-code platforms in organizations? Could you explain this?

#### **Low-code platforms in a federated model**

*With these questions, the federated way of working is linked with low-code platforms. From a low-code platform perspective, there will be looked at a federated model. The questions will answer what challenges and limits these platforms introduce and how they could work together. Furthermore, more enterprise-related questions will be raised how they, as low-code platforms, look at governance models, processes that rise in this low-code federated way of working.*

8. Do you have experience with low-code platforms in a federated model? If yes, could you elaborate on this?
9. How should central teams govern federated development teams in a low-code platform?
10. How can organizations make sure these new developers are guided through a process and comply with codes of conduct of low-code platforms?
11. What direction is your low-code platform going related to the rise of federated models in enterprises?

#### **Mechanisms for low-code platform implementations in federated models**

*First, a general view on low-code platforms in a federated model is discussed. With the following questions, I want to let the interviewee think about what type of mechanisms would improve these low-code platforms implemented in a federated model. The identified governance structures or processes from the questions above could be supported with mechanisms that will guide or force federated teams to improve the way of working.*

12. What mechanisms do you suggest that will support a federated model for low-code developments looking from a platform perspective?
13. How can you make sure with low-code platforms that federated teams will develop high-quality applications or processes quality of a high level?

#### **Final question**

14. Are there other important things that you think we should discuss that were not part of this interview?

## B. Research protocol federated model

### General introduction

Hi,

For my master thesis at the Eindhoven University of Technology, I am researching how enterprises can guide and govern federated development teams under a low-code paradigm. More specifically, nowadays, enterprises have to speed up with digital transformations and move to federated models. At the same time, low-code and no-code application software are more used than ever. Furthermore, RPA technology platforms such as UiPath and BluePrism are becoming easier to use. This interview's primary goal is to understand the variations in these federated low-code models in terms of governance and processes, what decisions can be made, and the underlying reasons.

I kindly ask you to read the questions beforehand. To have a more concrete and compelling conversation, I would like to invite you to think about the federated model implemented in your organization, the structures, and how the organization guides and governs these federated teams.

Across enterprises, there are various ways to describe a federated model. At the beginning of the interview, the definition used in this thesis will be explained so that the interviewee and interviewer are aligned. The federated model can be seen as a structure whereby autonomous development teams can share information and knowledge, but a central team coordinates them. Examples of this definition could also be a Center of Excellence, PaaS, or GSD.

The interview will be conducted virtually via a Microsoft Teams meeting (preferably a video conversation) because of COVID-19 circumstances. All personal and company-related information is confidential, and the interview will be anonymized. A transcript of the conducted interviews will be sent after the interview to confirm the recorded data. The gathered information will be used as input for my master thesis for the Eindhoven University of Technology.

Thank you,

Christophe Slangen

### Questions for the federated model

*[Explanation of the federated model to be aligned with the interviewee; the interviewer and interviewee should have a similar definition when talking about a subject before starting the interview.]*

#### General:

*With these questions, the goal is to receive a general overview and background information of the interviewee and the position he/she is in to ask more specific and detailed questions later in the interview.*

15. Could you tell me something about your background and your role in the federated model?
16. What platforms do you use/support, and can you tell me something about the federated structure you are using?
17. In what ways do you think low-code platforms change the way how these federated development teams are set up?

### **Federated Model**

*With these questions, I am interested in how their federated model looks like, what governance structure they have, what policies and collaboration between the central and federated teams there is, and how their security is ensured.*

18. What are the responsibilities and tasks for the central team and the federated team in your federated model?
19. What do you think are the most critical aspects of this federated model under a low-code paradigm?
20. What kind of collaboration is there between the federated teams and the central team?
21. How does your model look like in terms of:
  - The security process?
  - The process of becoming a federated team
  - Operations of the developed applications/processes
  - Going into the production process
22. How do you balance between autonomy of federated teams while having control over them as a central team?

### **Development process**

*With these questions to gather detailed information on the process that federated teams should follow in their model. What decisions they took and why. From this, mechanisms can be identified too, and a detailed process overview can be identified.*

23. How does the development process work for a federated team, and how are the teams guided through this process?

### **Mechanisms in model**

*First, a general view on low-code platforms in a federated model is discussed. With the following questions, I want to let the interviewee think about what type of mechanisms would improve these low-code platforms implemented in a federated model. How do they look at decisions that enterprises have to make for operations, security, governance models, processes etc.?*

24. How can central teams in federated models keep track and manage all projects within an organization?
25. What mechanisms do you think that can be used to improve and support a low-code federated model?
26. How can you make sure that the federated teams will deliver applications or processes of high quality?

### **Final questions**

27. What are the bottlenecks or limitations in your model?
28. Are there other things that are important in this model that we have not discussed?

## Appendix IX

### Coding scheme SRQ3

**Table 22:** Coding scheme relation federated governance and low-code development

Nr.	Name	Description	Files	References
<b>Opportunities &amp; chances</b>				
1.	Easier understanding, reviewing and collaborate with applications	Because low-code applications have a clear visual overview of an application, central teams could help more easily	4	8
2.	Bigger talent pool and local knowledge	Because low-code introduces a new type of users, full-stack low-code developers can be sought within the company	4	6
2.1	New developers in all segments & cross-functional teams	New developers in different segments and domains of the enterprise allow	2	4
2.2	No full-stack developer or long training needed	The group of developers that can build software applications becomes more extensive because of the ease of use and quick learning curves that LCAP provide	4	9
3.	Monitoring and access within the platform allows for central control	Monitoring applications in the IDE and gate access to data platforms and managing authorizations and roles for access can be managed from the platform	4	8
4.	Quality is already built-in in some building blocks	Because the main development in low-code development is done through drag and drop building blocks, a level of quality and security is already built-in	2	4
5.	Using DevOps and ease of deployment in low-code in local teams	Having business and operations in one team	7	15
<b>Risks and limitations</b>				
6.	Compliance and security	Risks regarding security and compliance issues	3	5
7.	Easy implementation but the need for adaptation of new WoW	Implementation of low-code platforms also requires an adaption of a new WoW	5	15
8.	Lack of expertise	The users' expertise could lack even more in a federated context	3	3
8.8	Only works with the right level of people		5	8

### Coding scheme SRQ4 - Governance dependencies

**Table 23:** Coding scheme dependencies

Nr.	Name	Description	Files	Ref.
1.	Maturity of team	The level of experience that a team has	7	21
1.2	Type of users	Some users do not have software development experience	7	10
1.2.1	# of years of experience	Amount of years that developers are developing software	3	5
1.2.2	Background	The type of background of developers	4	4



Nr.	Name	Description	Files	Ref.	
1.2.3		Certifications	The certificates that developers have	4	8
1.2.4		Followed courses internally	Amount of internal training followed with certificates	1	2
2.	Type and culture of an organization		The culture could influence the governance strictness of the mechanisms		
2.1	Operating model		The type of organization that prefers to manage operations centrally or have an agile DevOps culture	5	13
2.2	Culture		The focus and culture of the company, whether there is an entrepreneurial mindset or more conservative	6	15
2.3	Focus on compliance		The focus of institutions regarding compliance. E.g. Financial institutes are more focussed on compliance risks	3	6
3	Type of application				
3.1	Complexity of application		The level of complexity that an application has expressed in runtime, size, development cycles, and type of data	4	5
3.1.1		Development cycle	The times needed for each application cycle	2	3
3.1.2		Roles and multiple logics	The roles defined and logic of an application	1	1
3.1.3		Runtime and criticality of the application	The application could be an application that should run 24/7 and need day & night support, or an application could run only during working hours	5	13
3.1.4		Size of application (users and volume)	Personal use or multiple users or organizational-wide use	3	10
3.1.5		Type of Data and Data model and access to integrations	Types of data used in an application	7	17
3.2	Internal-external facing		If the application is going to be used by internal people only or also outside of the company	3	6
4.	Type of technology		The technology platform used has a set of features and characteristics	5	6

**Coding scheme SRQ4 - Mechanisms and tasks for a central team**

Please note, the codes could be called differently in the study. Furthermore, the identified mechanisms in the coding scheme are marked with an M followed by the mechanism.

**Table 24:** Coding scheme mechanisms and tasks

Nr.	Category			Description	Files	Ref.
1.	C- Assisting and advising			Retrieved from SRQ2		
1.1		Coach, support and assist the federated teams		Assisting and helping the federated teams or their developers in the developments or guiding through a process	8	30
1.1.1			M- A SPOC from the central team	A model where there is somebody from the central team that is in the same domain that can guide the team with the application	3	7
1.2		Support agile way of working		In a low-code environment, the agile way of working is a commonly used method	5	12
2.	C- Communication and alignment			Retrieved from SRQ2		
2.1		Align federated teams		Bringing federated teams together or let them know about each other or the central team and its processes	7	9
2.2		M – An internal community		An internal community with all developers of a company and maintaining this	9	15
2.2.1			M- Hackathons or boot camps	Organizing hackathons, boot camps or other activities where developers from multiple federated teams are invited	2	3
2.3		M-Regular meetings		Having regular meetings with federated teams together	5	5

Nr.	Category	Description	Files	Ref.
2.4	Promoting the platform	Promoting the development and technology in the enterprise	5	11
3.	C- Defining strategy, operating model & federated team responsibilities	Retrieved from SRQ2		
3.1	Having a low-code strategy	A strategy for the implementation and roadmap of the technology within the company and the federated model	4	10
3.1.1	Aligning key stakeholders	Align architects, federated teams, but also higher management on the low-code program	3	5
3.1.2	M- Change-management program	Making sure that stakeholders are ready for the low-code federated program	3	5
3.1.3	Focus on people	Focus on the right level of people within an organization	4	5
3.1.4	Portfolio	Having an overview of the whole portfolio of applications within the company	4	6
3.1.5	Strategy for growing as a platform	Having a growth strategy for the low-code platform and federated model within the company	5	10
3.2	Responsibility federated team	The responsibilities that a federated team should have	1	1
3.2.1	Team composition	A decision by the central team about the roles within the federated team	1	1

Nr.	Category			Description	Files	Ref.	
3.2.1.1				Have a business analyst	A business analyst is somebody that can identify opportunities in a business unit or context	2	2
3.2.1.2				Have a delivery manager	Decides on which applications are going to be built, which are going to deploy and which are going to be decommissioned	1	1
3.2.1.3				Have a tester in the team	A role who can test applications and review them properly	3	4
3.2.1.4				Have an expert developer in the federated team	An expert developer with experience on the platform	6	26
3.2.1.5				Have multiple people in a fed team	Requirement for federated teams to have multiple people in the team	4	9
3.2.1.6				Having a tech lead or product owner	A tech lead or a product owner is a role that oversees the team and has a form of technical knowledge	3	6
3.2.2			Evaluate measurable value created		Measuring how much value an application adds	1	1
3.2.3			Prioritization of projects or applications		Prioritizing applications and setting timelines	2	3
3.2.4			Ensure quality of the application		The quality of an application regarding functional requirements	3	3
3.2.5			Responsibility to grow to be self-sustainable		To have a strategy with goals to be able to work self-sustainable and autonomous	5	5

Nr.	Category	Description	Files	Ref.	
3.2.6		Security built-in	A level of security that protects the application	3	3
4.	C- Facilitating and maintaining technologies		Retrieved from SRQ2		
4.1		Automating the development process	By using self-service API's	6	15
4.1.1		M- A CI/CD pipeline	A CI/CD pipeline with a test framework can ensure automatic testing and deploy for applications	3	5
4.2		Manage talent pool for low-code centrally	Managing the pool of new developers that can build in the platform	1	1
4.3		Managing cloud infrastructure and platform	Maintaining the cloud infrastructure	7	13
4.4		Managing integrations	Manage integrations for the low-code applications	3	5
4.5		Managing security	Managing security of the platform	6	13
5.	C- Knowledge sharing and educating		Retrieved from SRQ2	0	0
5.1		Educating	Educating employees to increase knowledge	6	6
5.1.1		M - Explaining on development process and platform	Explaining the development process, policies and explanation of the platform	10	36
5.1.1.1			M- Manual Having a manual where the process and policies are explained	3	8
5.2		M - Training	Providing training	7	21

Nr.	Category			Description	Files	Ref.
5.2.1			M- A certification program	Having a program with various certificates for the federated teams to obtain certificates and grow	2	4
5.2.2			Provide internal training	Training on ways of working, compliance and security aspects or brand identity involvement in the platform	3	5
5.3		Share knowledge between federated teams		Sharing knowledge and information between federated teams and between central and federated teams to align	6	11
6.	C- Maintaining and managing reusable assets			Retrieved from SRQ2	4	7
6.1		Creating templates		Templates for federated teams to make use of pre-built UI/UX formats	2	2
6.1.1			Creating best practices	Providing best practices on technical development methods or implementing internal processes	3	3
6.1.2			UI-UX	User interface and user experience	4	10
6.2		Identify applicable com components		Identifying which reusable components should be adopted	4	5
6.3		Maintaining common components and app store		Maintaining and updating the components that are or can be used by other federated teams	7	17
6.3.1			M- App store	Maintaining a library with developed components or applications with explanations on the components	6	9

Nr.	Category	Description	Files	Ref.		
6.4		Standardizing reusable components from federated team	The central team should standardize a component or application when it is decided that it will be reused	4	4	
7.	C- Monitoring assess and evaluate			Retrieved from SRQ2	0	0
7.1		Assess maturity level of federated teams	Assessing and determining the level of maturity of federated teams. This can be done according to (M-) a maturity and complexity framework	4	9	
7.2		Measure value out of the federated model	Measuring by metrics such as ROI or use another KPI to track the value of the federated program	3	7	
7.3		Monitoring federated teams	Monitoring teams and their status	7	15	
7.3.1		M- A dashboard	A dashboard to monitor the progress of applications in federated teams	4	6	
7.3.2		Monitor application progress	Monitor the progress of applications developed by federated teams	6	13	
7.3.2.1		M- Walk along checks	Have somebody looking into the application from time to time to guide them and assess and monitor the application	4	10	
7.4		Review and evaluate federated applications	Evaluating the applications developed by the federated team	1	2	
7.4.1		Identifying risks	Find risks involved in the developments of applications of federated teams	7	12	

Nr.	Category			Description	Files	Ref.	
7.4.2			M - Security or risk assessment		An assessment to understand the level of risk	6	15
7.4.3			M- Code reviews		Do code review checks to see and review vulnerabilities of applications	7	19
7.4.5			M- Spot checks		Have spot checks during the development to review the application	5	8
7.4.6			Testing before deployment		Before the application goes into production, have the application tested	6	12
7.4.6.1				Compliance & security check	A formal compliance and security check on the application	3	10
7.4.6.2				M- Access test	Access test to review the authorization quality of the application	2	3
7.4.6.3				M- Penetration test	Do PEN-tests to review at what point the application breaks and what its performance is	3	4
7.4.7			Understand federated team project		Understanding what the project is about and advising on how to approach it	7	24
8.	C- Setting development guardrails and organizational policy				Retrieved from SRQ2		
8.1		M - Build testing capability and framework			Having a test framework that shows how applications should be tested and what measures are sufficient	6	11
8.2		Ensuring design freedom and autonomy to			The autonomy to design and feel free to not being controlled	6	25



Nr.	Category	Description			Files	Ref.	
		federated teams					
8.3		Processes in place			Having policies in place and define processes that teams should follow	1	2
8.3.1			Decide on the deployment process		Having a centrally decided deployment process in place that the federated teams should follow	7	19
8.3.2			Establish development process with guidelines		Have a development process with guidelines for the federated teams to make clear what is expected and what has to be followed	8	35
8.3.3			M - Explanation and transparency of the development process		Define this development process as an artefact		
8.3.3.1				Design stage	The design stage where the requirements and preparations for developments take place	4	10
8.3.3.2				Development & testing stage	The development & testing stage where the actual developments and testing of the application take place	2	2
8.3.3.3				Identification & Awareness stage	The first stage of the process where the application is identified and starts	3	5
8.3.3.4				Intake stage	After the identification, an intake should take place that monitors and captures what is identified	4	7
8.3.3.5				Pre-deployment stage	The critical stage before the application goes into production.	5	12

Nr.	Category			Description	Files	Ref.
8.3.3.6				Support and production stage After the development, the application will be placed in the support and production stage, where it should be monitored	1	1
8.4	Providing governance structure			Providing a clear governance structure to the federated teams how the central team works and what the federated model is	4	7
8.5	Set the architectural framework			Having an architectural overview of federation and architecture framework of applications	5	14

## Appendix X

### A. Explanation and transparency of the development process

Below the intranet pages of the explanation and transparency of the development process artefact are shown. First, the team page is displayed with information on the team and developed products. In this way, the identification and awareness of teams are stimulated, as explained. Secondly, on this page, an introduction of the federated model with explanations on working and the responsibilities is shown. To better understand how the teams should develop, the page visitors can click on the development way of working. The goal of this explanation was to have a guided pathway where the federated could go to understand what they have to do at what point. To give an idea of how this looks like, one page is shown. In these pages, extra information regarding that stage is provided to the federated teams.

#### *Central intelligent automation teams*

Since the central team should advise the teams and other business units on the type of technology they should use when a project is realized, they should find a holistic overview of all technologies provided. Therefore, the awareness stage of the designed process (as shown in *Figure 19*) was translated into an intranet landing page with explanations on the platform, federated model and examples with videos. On this page, the intake process is also shown as well as information on the federated model. To make sure the process is being updated and used, a guidebook including how to update and change this guided pathway and how to improve it, including all files, is attached to it as well. A screenshot of landing page can be found below.

*-This figure is removed due to confidentiality reasons-*

**Figure 32:** Screenshot 1 - landing page for promotion and transparency intelligent automation team

*-This figure is removed due to confidentiality reasons-*

**Figure 33:** Screenshot 2 - landing page for promotion and transparency intelligent automation team

*-This figure is removed due to confidentiality reasons-*

**Figure 34:** Screenshot 3 - landing page for promotion and transparency intelligent automation team including a link to the intake process

*-This figure is removed due to confidentiality reasons-*

**Figure 35:** Screenshot 4 landings page for promotion and transparency intelligent automation team

#### *Central low-code team landing page*

There is a link from the central intelligent automation team that provides more information on the low-code team. This is the landing page to redirect federated teams to details on the federated model, development process, shows developed use cases, and become a community member.

*-This figure is removed due to confidentiality reasons-*

**Figure 36:** Transparency on the central team

*-This figure is removed due to confidentiality reasons-*

**Figure 37:** Explanation of the central team, technology and pre-built applications

*-This figure is removed due to confidentiality reasons-*

**Figure 38:** Presentation on community meetings internal channel

### *Development process*

In this process, the first two phases are cut off because it only focusses on the development part. The intake process is already conducted in an earlier phase. In this picture, the user can click on a stage in the process and get more information about the stage, including the federated teams' responsibilities.

*-This figure is removed due to confidentiality reasons-*

**Figure 39:** Screenshot of interactive guidance through the development process

### *Design stage*

When, for example, clicking on the design stage, the user will be redirected to another page with more information and actions that have to be conducted in this stage. This page can be found below.

*-This figure is removed due to confidentiality reasons-*

**Figure 40:** Screenshot explanation page design stage process

*Information on the federated model*

As identified, the federated teams should understand what responsibilities they will have when setting up a federated team. To have effective governance, the governance framework should be transparent and open (Weill & Ross, 2004a). Below, the federated structure is explained, and the suggested process is presented. In this process, it is also possible to click to receive more information on each stakeholder's process and responsibilities. As can be seen, the developed method is also used to explain the way of working.

*-This figure is removed due to confidentiality reasons-*

*Figure 41: Screenshot explanation on governance*

## B. An intake process

The intake process consists of an intake form with questions that will give the central team a holistic view of what the requester wants to realize. This form is created both for central development as well as federated development. First, the process is shown how the questions were formulated and identified and next, the intake process is shown.

### 1. Questions

During the design of this process, stakeholders from the three teams were asked to give input. Next, the security officer and architect were asked to give input. In this way, a holistic overview of relevant questions was generated. The form is automated, and some questions have multiple levels; if question A is answered with B, then question C will be asked. Similarly, if question A is answered with D, then question D will be asked. The questions are iteratively added and adjusted; these were documented and are shown in *Figure 42*. When all questions were identified, these were added to a software development tool. This tool allows users to quickly build questionnaires or, in this case, intake forms.

*-This figure is removed due to confidentiality reasons-*

**Figure 42:** Questions intake form

*-This figure is removed due to confidentiality reasons-*

**Figure 43:** Questions intake form 2

*-This figure is removed due to confidentiality reasons-*

**Figure 44:** Questions intake form 3

*-This figure is removed due to confidentiality reasons-*

**Figure 45:** Developed internal intake form

### 2. Process

To make sure that all stakeholders that will make use of the process are aligned, the process is developed in more detail. Furthermore, an presentation to all stakeholders is provided to answer questions related to the new process and let them adapt to the new way of working. The process is also automated so that the business analysts from the central team receive the filled-in intake form of the requester and receive a notification of this. This process is automated with a no-code software tool. First, the researcher explored this tool, and later, this process is developed iteratively.

*-This figure is removed due to confidentiality reasons-*

**Figure 46:** Screenshot back-end code for the intake process

*-This figure is removed due to confidentiality reasons-*

**Figure 47:** Screenshot intake form confirmation

The process divides the request into two sub intakes; a problem intake and a solution intake. In this way, the questions are adjusted to the requester. The automated process sends an email to the requester

and the central team stakeholders to let them know that the intake was successful and let the central team know that there is a new incoming intake.

*-This figure is removed due to confidentiality reasons-*

**Figure 48:** Screenshot of result emails from automated intake process for requester and central team

*-This figure is removed due to confidentiality reasons-*

**Figure 49:** Screenshot of notification feature for intake process for the central team

### Dashboard

When an intake is submitted, the output of the intake was pushed to a dashboard. A card was created with all details and stamps in this dashboard and automatically placed in the first intake handling column. When the intakes were reviewed, they were dragged to the next column, 'validation'. After the central team decided which team would handle the intake, the card was placed in that dedicated column. Furthermore, some intakes were on hold since the federated team did not want to not focus on this project or when there was, for example, no budget. The cards were labelled with various tags. These tags could include, for example, 'central development', 'federated development', 'high risk' or 'external facing'. In the dashboards, these tags could be filtered. In this way, a clear overview of high-risk federated development projects could be shown. In addition, the priority of the project could be attached to each card (see *Figure 52*). Furthermore, to ensure that the central team had a clear overview of who was responsible for which project, names were attached to the card (see *Figure 53*). This was important when the program owner or the central security owner wanted to know more about a project. Therefore, the SPOC of the central team was linked to the card. In this way, a clear overview is developed where SPOC's of the central team were assigned to projects of federated teams. Moreover, this dashboard could be showed internally to show the progress and demand of the whole enterprise. This could inspire other federated teams to develop new applications or to reuse and implement projects that were being developed.

When a project is finished, the central team can click on complete. Then, the project is not shown anymore on the dashboard but will still be counted as a project. In this way, it is possible to click on the chart button and see the list of finished intakes. Furthermore, the charts will show how many intakes are submitted, how many are finished, how many are completed, stopped or placed on hold. In this way, the central team will have a clear overview of the demand and the value of the federated program. This can be used as a KPI monitor to track the progress (see *Figure 54*).

*-This figure is removed due to confidentiality reasons-*

**Figure 50:** Intake dashboard for handling all incoming intakes

*-This figure is removed due to confidentiality reasons-*

**Figure 51:** Screenshot of a card in the dashboard with details of intake

*-This figure is removed due to confidentiality reasons-*

**Figure 52:** Screenshot of priority levels of projects in intake card within the dashboard

*-This figure is removed due to confidentiality reasons-*

**Figure 53:** Screenshot of assigning team members to projects

*-This figure is removed due to confidentiality reasons-*

**Figure 54:** Screenshot of charts for the intake process

#### *Manual process*

With the implementation of the intake process, a manual process is defined to ensure that the intake process is adopted. Therefore, the intake process is zoomed in. As can be seen, also the technology selection is incorporated in this process since the intake process was also applicable for the other two technologies. In addition, monitoring happens using the dashboard as described before.

*-This figure is removed due to confidentiality reasons-*

**Figure 55:** Extended intake process explanation to stakeholders

*-This figure is removed due to confidentiality reasons-*

**Figure 56:** Stakeholder alignment and change management session for artefact



### C. An internal community

A community was created to gather all federated teams together and to share knowledge. Therefore, first, all teams and a list of all developers that were onboarded and actively developing on the low-code platform were approached. These internal developers were added to a common channel and updated frequently. Communication about outages on the platform, new updates, and upcoming events or community meetings was, next to emails, placed on this channel. The focus on this was to create an open and informal community where users felt confident to ask questions. Furthermore, it was easy to become a member of this community, and information was also available, as can be seen in *Figure 38* and *Figure 58*. In addition, after this, regular community meetings were organized by the researcher. In advance, a list of topics was selected that could be presented during these meetings. In this way, the topics could also educate the federated teams on crucial topics such as SSO implementation. There was room to ask questions to expert developers from either the central team or other federated teams during the meetings. A standardized format for the presentations was created to achieve a professional image. The federated teams attended the meetings, and sometimes the presentations were presented by a federated team that talked about their developed products. During the meeting, the researcher acted as a mediator, as explained in *section 5.3-A*, to align the federated teams during the sessions. The person that presented the topic varied each time. Each session was recorded, and after the community meeting, the recording was uploaded to the team's channel and directly pushed to the Sharepoint page. In this way, the central team's Sharepoint became the hub to find all low-code related topics for the company. Furthermore, automation was created that copied the final presentation to an open and accessible location for the federated teams to be able to download the slide decks.

*-This figure is removed due to confidentiality reasons-*

*Figure 57: Screenshot community meeting sessions*

*-This figure is removed due to confidentiality reasons-*

*Figure 58: Communication about internal community and meetings*

*-This figure is removed due to confidentiality reasons-*

*Figure 59: Example of promotion messages for community meetings to align federated teams*

**D. Security self-assessment**

The security assessment was developed to identify the risks of the application that the federated team will develop and educating them on the security measures. Together with two security officers, a classification matrix was developed on the application level. This matrix divided an application that was being developed into one of three different levels of a set of categories. Based on the score of the assessment, a set of actions had to be followed. Therefore, the first table indicated what each level means within that category, and the second table advises what measures should be taken on the application level for low-code development for the case company. The different categories were (i) interfaces, (ii) data, (iii) confidentiality, (iv) data integrity, (v) financial transaction or segregation of duties, and lastly, (vi) data availability.

**Table 25:** Security classification matrix for self-assessment

<b>Matrix of Classifications</b>			
<b>Category</b>	<b>1 - Low</b>	<b>2 - Medium</b>	<b>3 - High</b>
<b>Interfaces</b>			
<b>Data Confidentiality</b>			
<b>Data Integrity</b>			
<b>Financial Transaction or Segregation of Duties</b>			
<b>Data Availability</b>			

*-This information is left out due to confidentiality reasons-*

**Table 26:** Security measures matrix for security self-assessment

<b>Matrix of Security Measures</b>			
<b>Category</b>	<b>1 - Low</b>	<b>2 - Medium</b>	<b>3 - High</b>
<b>BASIC requirements</b>			
<b>Interfaces</b>			
<b>Data Confidentiality</b>			
<b>Data Integrity</b>			
<b>Financial Transaction or Segregation of Duties</b>			
<b>Data Availability</b>			

*-This information is left out due to confidentiality reasons-*

### *Self-assessment*

A self-assessment was created for federated teams to automatically identify what measures should be taken into account by the federated teams based on the created matrixes. Furthermore, the central team's security officer wanted to monitor the developments of a federated team still for audit purposes and to be able to look back on the security levels. Therefore, the results of the assessments were stored in a central place. This overview of results was shown on an internally accessible page that was only available for the central and security officer members. This can be seen in *Figure 64*. In the dashboard, there was an option to put comments to the intake, change the status and see if the requester wanted to go into production or not. Based on this output, the central security officer could review the incoming security assessments and process them accordingly. The security officer could change the status to 'ready for production, so the federated team would get notified that the security part is accepted. Therefore, this mechanism ensured that the federated team and the security officer did not have to wait for a manual meeting for approval in some cases and that the design freedom was ensured as much as possible.

*-This figure is removed due to confidentiality reasons-*

**Figure 60:** Automatic reply to security self-assessment requester showing the results in Microsoft Teams

*-This figure is removed due to confidentiality reasons-*

**Figure 61:** Screenshot of back-end coding process security self-assessment

*-This figure is removed due to confidentiality reasons-*

**Figure 62:** Screenshot 2 of back-end coding process security self-assessment

*-This figure is removed due to confidentiality reasons-*

**Figure 63:** Screenshot 3 of back-end coding process security self-assessment

*-This figure is removed due to confidentiality reasons-*

**Figure 64:** Screenshot of the central dashboard of security-assessments

*-This figure is removed due to confidentiality reasons-*

**Figure 65:** Automatic reply to security self-assessment requester showing the results in email

*-This figure is removed due to confidentiality reasons-*

**Figure 66:** Automatic reply to the security officer and central team showing the results in email