Agile Architecture in Research Platforms: A Case Study of TSD

Walid Haidari

Thesis submitted for the degree of
Master of Informatics: Programming and System Architecture
60 credits

Department of Informatics
Faculty of Mathematics and Natural Sciences

UNIVERSITY OF OSLO

November 2020

Abstract

Research in the digital age is now meeting new opportunities in which technology allows further use of research data. In platform literature, the rise of consumer platforms is among the popular themes to discuss. But non-consumer oriented platforms such as research platforms rarely get mentioned in the literature. In addition to facilitating platform growth in terms of user relations and service development, research platforms have to be a safe space to store research data. The focus on security in research platforms might halter innovation by having greater control of what is shared with the users of the platform.

Bloomberg (2013) presents the concept of Agile Architecture, which has not been discussed in the current system architecture literature. As there are no de facto standards of what an agile (system) architecture is, we shed light on Bloomberg (2013) to present how an agile architecture can be in a research platform.

This thesis presents a case study that explores the characteristics of a research platform named TSD. The platform is used by the University of Oslo and partnered institutes around Norway. What makes TSD unique among platforms is that the platform continually develops and offers services, even though they meet challenges because of security in the sensitive research data they store and use in services. Storing sensitive data can be somewhat more complicated than storing non-sensitive data due to the need to follow laws and regulations. Qualitative data was gathered via interviews and documentation in the period between 2019 to 2020.

Through a framework that we created, using concepts from Bloomberg (2013), empirical evidence about the TSD platform was categorized and analyzed. The results were divided into Processes, Services, and Loose Coupling in the architecture. They indicated that the modular and layered architecture of TSD allowed its services and processes to become more flexible, enhancing the notion of business agility, which Bloomberg (2013) presents. A major architectural component that enables TSD to be agile and secure at the same time is the implementation of SAPEIN. SAPEIN is a secure API architecture used to replace their integration-heavy IAM named Cerebrum. The results also indicated that a platform foundation served to promote service development using resources provided by the platform owner, further strengthening the agility of the system. A modular and layered architecture led to less interdependence between components in the system, making the system respond

efficiently to changes. This type of architecture also focused on service flexibility, which could be seen mostly in reusing existing services. In this thesis, we contribute to the existing literature by investigating the fundamentals of agile architecture in a research setting and establishing insight in what research platforms and agile architectures are.

Keywords: Research Platform, Digital Platform, Agile Architecture, System Architecture, Loose Coupling, Modularity

Acknowledgements

Writing this thesis has been both insightful, exciting and challenging, and I would like to take a moment and thank everyone who have helped me on the way.

Firstly, I am thankful for my supervisor Egil Øvrelid, for the guidance, discussions and new insights in this thesis. Additionally, I am thankful for the help and cooperation of the TSD team. Especially Gard Thomassen for valuable knowledge and useful information during our talks.

Finally, I would like to thank my friends and family for all the support and love through these years. Without each one of you none of this would be possible. Special thanks goes to my dear mother who is always there to encourage and comfort me, and to my late father whom I dearly miss, for always inspiring me to do my best.

Walid Haidari
Oslo, 2020

Contents

List of Figures

List of Tables

Abbreviations

- AA - Agile Architecture
- API - Application Programming Interface
- BPM - Business Process Management
- EA - Enterprise Architecture
- ESB - Enterprise Service Bus
- EU - European Union
- IaaS - Infrastructure as a Service
- IAM - Identity and Access Management
- ISO - International Organization for Standardization
- IT - Information Technology
- NSD - Norwegian Center for Research Data
- PaaS - Platform as a Service
- REST - Representational State Transfer
- SaaS - Software as a Service
- SAPEIN - Secure API for eInfrastructures
- SOA - Service Oriented Architecture
- TSD - Services for Sensitive Data
- USIT - University Center for Information Technology
- UiO - University of Oslo

Chapter 1

Introduction

Research in the modern digital age utilizes computers and software increasingly more often with each year. This digitization of the research field creates an increasing demand for environments that facilitate storage and research data usage. The Centre for Information Technology at the University of Oslo recognised this need and demand from the researchers that required a common place to store their data. As a result, Services for Sensitive Data was created and allowed research groups to use it as a storage and processing facility for their data. TSD underwent a rapid growth in popularity which continues to this day. As of such, the organisation has been presented with a range of emergent challenges. The system expanded and shifted towards a platform architecture to scale with the growing user base and their need for constant service development.

This thesis will address both practical and theoretical issues. The practical issues concerns activities related to problems regarding storing, sharing, and research on data. Research data is often spread around, making it hard to keep safe and even harder to share, resulting in research projects taking longer to finish. Furthermore, in our modern time, research projects are expected to be established swiftly. For these practical issues, platform fundaments in research platforms can help store research data securely and establish research projects fast.

The theoretical issues concerning this thesis relate to how one might establish secure and efficient storage structures. These structures, which come in the form of platforms in this thesis, require an agile approach. When talking about what is agile, one often thinks about agile approaches to software development, but this thesis seeks to explore agility withing the systems architecture.

Platforms such as research platforms are scarcely represented in the literature. The focus has been on the emergence of consumer-oriented platforms such as the ones Tiwana (2014) presents. A practical example of a research platform is the TSD system. While platforms such as those in Tiwana (2014) focus on values created by consumers and providers within a platform ecosystem, platforms such as TSD focus on values created by and for the researchers.

Furthermore, Bloomberg (2013) presents in their book the notion of Agile Architecture. This topic is rarely discussed in the literature on system architecture. One reason is that there are no formal definitions of the term agile architecture. Within a research platform, security might come off as a hindrance for the agility in the architecture. The tension between security and flexibility often occurs in systems and platform architecture literature.TSD continuously develop new services and establish new research projects on demand. This was not always the case, as they stumbled upon challenges throughout the years. Recently, the TSD team designed SAPEIN, which is a secure architecture that also enables flexibility in the system. Hence, the thesis's scope will focus on agility within platform architecture, especially research platforms. Theory from Bloomberg (2013) will be used to get a better understanding of agility and a framework used to collect the right evidence.

The research method used in this thesis is qualitative research. Through a case study, we gather information about the complex phenomenon of the TSD platform. Using the empirical evidence gathered from the case study, we answer the research question regarding agile architecture in research platforms.

1.1 Personal Motivation

Through the courses I have taken during my master's degree, the courses regarding Information Systems and Platform Ecosystems made me curious about their topics. This thesis is written for the research group Digitalization and Entrepreneurship. Bloomberg (2013) presents in their book the concept of agile architecture, a different definition of architecture focused on principles of agility. However, this is only discussed widely among enterprises and how they should make their businesses respond to changes efficiently. TSD fits into numerous concepts mentioned and became an interesting subject for a study on agile platform architecture within research.

1.2 Research Question

The research question we aim to answer based on the scope of this thesis is as following:

How can we establish an agile architecture in a research platform?

1.3 Contributions

This thesis contributes to the field of information systems and system architecture by presenting and discussing the notions of research platforms and agile architecture. We also seek to contribute to the gap in the literature of

agile architecture in research platforms, We will use a combination of empirical evidence from a case study and relevant literature to so.

1.4 Thesis Structure

The remaining chapters of the thesis are as follows:

- **Chapter 2: Related Literature** introduces relevant literature regarding digital platforms, system architecture and information security.

- **Chapter 3: Theoretical Framework** creates a framework in which the empirical evidences will be examined based on the literature from Bloomberg (2013).

- **Chapter 4: Methodology** discusses the choice of research method used to conduct the study in this thesis. The chapter also presents ethical considerations when conducting research.

- **Chapter 5: Case** gives a short overview over the context of the case study and presents the organisation.

- **Chapter 6: Findings** presents the results of the research from the case study. These are presented in an order that corresponds to the scope.

- **Chapter 7: Discussion** discusses the empirical evidence in relation to relevant literature, to answer the research questions. Finally, we mention limitations to the study.

- **Chapter 8: Conclusion** concludes the discussion and answers the research question. Proposes future work at the end of the chapter.

Chapter 2

Related literature

This chapter will go further into the platform- and system architecture theory to create a foundation for the research we will do in this project. The section about Platforms will contain definitions of the concept and discusses the concept of research platforms. The section about system architecture will introduce this broad topic and include three methodologies used within the field and related to the thesis' research subject.

We have chosen to present platform theory as the thesis revolves around a research platform. The platform section also seeks to explain API, digital debt and options because of their importance in managing digital platforms. As the term "research platform" does not have much presence in platform literature, we present literature that mentions this. As agile architecture is one of the main topics of this thesis, we provide a better understanding of agile architecture, methods, and the various techniques used within the field. As the case study revolves around a system that focuses on sensitive research data, we also present information security and privacy at the end.

2.1 Platform

A platform in basic terms classifies as a structure that other objects are placed upon (Dictionary, 2020b). This section will present and discuss platforms in the form of digital platforms. Social media platforms like Facebook and Twitter, or mobile platforms such as Apples' iOS platform has taken the world by storm by disrupting traditional industries (Tiwana, 2014; Reuver, Sørensen, and Basole, 2017).

IS literature divides platform research into different perspectives. Rolland, Mathiassen, and Rai (2018) divides the platform literature into Engineering-, Economic- and a third perspective that is the Organizational. The three perspectives on digital platforms are described as: engineering having a focus on the platform as a modular architecture with a stable programmed core with peripheral components revolving around it in; economic focusing platforms as disrupting markets that break with traditional markets by facilitating interactions between consumers and producers; organizational focusing on a layered modular architecture with a stable core and external changeable

components, that facilitates innovation within an organization.

"A software platform is a software-based product or service that serves as a foundation on which outside parties can build complementary products or services. A software platform is, therefore, an extensible software-based system that provides the core functionality shared by “apps” that interoperate with it, and the interfaces through which they interoperate” (Tiwana, 2014, p. 5).

Using this definition, one can explain platforms as systems that share their data through boundary resources, allowing third parties to use them to develop features, applications, or services. In Tiwana, 2014, the platform definition also focuses on the multi-sidedness of the system, pointing out that a system can only be a platform if two or more actors communicate with each other, for example, third-party app developers and end-users. Baldwin and Woodard (2008) shifts the focus on a platform away from its market heavy view of multi-sidedness and focuses more on an architectural view, pointing out that a system should be modular in order to be called a platform.

"Platform architectures are modularizations of complex systems in which certain components (the platform itself) remain stable, while others (the complements) are encouraged to vary in cross-section or over time. Among the most stable elements in a platform’s architecture are the modular interfaces that mediate between the platform and its complements" (Baldwin and Woodard, 2008, p.3).

2.1.1 Boundary resources

Both definitions by Baldwin and Woodard (2008) and Tiwana (2014) highlights complements of the platform and the interfaces. The complements of a platform can, for instance, be an application or a service. Application in this context relates to software applications and software services, which are seen as sub-systems connected to the platform (Tiwana, 2014).There needs to be interfaces to have these applications and platform core communicate and exchange information with each other. Interfaces work as a set of specifications that describes how platform and applications will operate with each other. A form of interface often used in platform literature is API (Application Programming Interface)(Tiwana, 2014; Reuver, Sørensen, and Basole, 2017; Rolland, Mathiassen, and Rai, 2018).

Application Programming Interface (API) is an interface programmed and designed by the owners of the source (code) platform that the application is going to be developed upon by the third-party developer. This set of specifications allows the developer to develop the application without knowing the complete source code (Rossen, 2019).

In relation to platforms, API allows one to many relations between the platform and the apps. As it enables application developers to use the capabilities of the platform without having to know how the platform works (Tiwana, 2014). The relation can be seen on the illustration presented in figure 2.1.

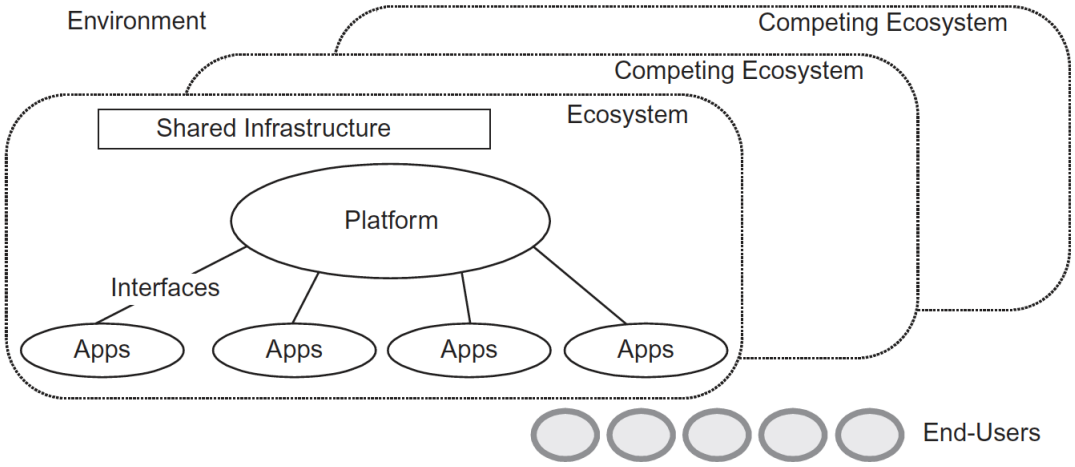


FIGURE 2.1: A Platform Ecosystem as presented in Tiwana 2014

In addition to defining digital platforms as technical artifacts similar to the descriptions above, the literature also defines it as a sociotechnical assemblage enveloping technical aspects of hardware and software and organizational processes and standards (Reuver, Sørensen, and Basole, 2017). The sociotechnical aspect of a platform emphasizes the concepts of multi-sidedness (as mentioned from Baldwin and Woodard (2008)) and Network effects. Network effects, or network externalities, can be seen as the degree to which the different actors affect the platform ecosystem by bringing value to the other users. Platform literature distinguishes between two forms of network effects and their positive or negative effects: Same-side and cross-side network effects. This shows that both architecture and network effects have a play in which ecosystems win over end-users and developers to their platform ecosystems.

2.1.2 Digital debt and options

In Rolland, Mathiassen, and Rai (2018), the terms digital debt and digital options are presented as a way to understand the complexity of an organization's digital platform. Digital debt refers to the inertia and path dependencies that an organization implements to their infrastructure and processes, while digital options refer to opportunities to invest in features that will increase an organization's value proposition (Rolland, Mathiassen, and Rai, 2018). The term digital debt originates from the more software programming related notion of Technical debt, which is when developers make decisions that would benefit them in the present, but resulting in costly long term complications (Martini, Sikander, and Madlani, 2017). The concept of digital debt might gather negative attention because of its relation to technical debt. However, Rolland, Mathiassen, and Rai (2018) argues that digital debt can be planted in the form of digital options, meaning that the interactions between digital options and digital debt can contribute to the generative of a platform. Thus, digital debt is neither negative nor positive, but rather a notion that needs to be managed in terms of options.

2.1.3 Research Platforms

Platform literature is relatively new and has been focusing more on commercial platforms, and platforms such as Research Platforms have not been investigated as much. The literature is lacking, but there have been studies regarding these platforms. These research platform studies tend to focus on research platforms that focuses on specific fields such as biology (Schindelin et al., 2012), energy and climate (Feng et al., 2017) or health information (Braa and Sahay, 2017). While some studies are focusing on specific research areas, we can see a common need for a platform that provides infrastructure which allows researchers to work and collaborate on research data (Matsubayashi and Kurata, 2017; Shin et al., 2019).

The European Commission, an institution of the EU, has launched the project OpenAIRE2020 (Shin et al., 2019), which works as an initiative to promote research collaboration across institutions within Europe. The European Commission seeks to make OpenAIRE a "hub for 3rd party providers to build innovative services that explore new forms of scholarly communication and promote alternative, competitive Open Access publishing models" (European-Commission, 2020a. This relates to the notion of digital platforms as a codebase with functionality shared by applications interoperating with it.

Through investigating these platforms, we find two characteristics that are present in research platforms:

- The infrastructure of a research platform needs to support research processes by utilizing research data (Shin et al., 2019; European-Commission, 2020a; Schindelin et al., 2012). Research data analysis-, manipulation-, provision-, monitoring tools and services supports this.

- Research data might contain sensitive and confidential information. Making security a essential part in distributing data in the platforms.

2.2 System Architecture

System architecture is a high-level model of a system, where it shows the connections and interactions between internal and external components. It is not the same as software architecture, which can be defined as an architecture describing the digital components and their interactions. A system is more than just components of the software, and therefore software architecture is just a part of systems architecture (Weber and Dustdar, 2012, p.3-4). There are no de facto standards or universal definition of what system architecture is and looks like, which means that the architecture is formed by the specific purpose it is supposed to fulfill. The following sub-sections will present architectural styles and frameworks that are prevalent in this thesis.

2.2.1 Platform Architecture

2.1 discussed digital platforms as a concept in IS literature. In this section, we will further discuss the architectural aspects of Platform Architecture. As mentioned in 2.1, a digital platform is conceptualized as a modular layered architecture (Baldwin and Woodard, 2008; Tiwana, 2014), with layers such as the core, its external components, and the ecosystem revolving around the platform. This can be illustrated as several separate rings representing the layers, with the platform's software core being in the center. Illustrating a platform like this might help to make it easy to understand, but a platform architecture is more complex than what meets the eye in such an illustration.

Modularisation is an important term in Platform literature, as platforms are modularisations of complex systems (Baldwin and Woodard, 2008). To get a better picture of what makes a system modular, one has to look at tight and loose coupling. Tight and loose coupling describes the dependency of elements in a system, as well as actors. When a system is tightly coupled, the elements are strongly dependent on each other with a high degree of dependency. In comparison, a loosely coupled system consists of elements that are independent of each other but still responsive. This makes it easier to make changes in different elements without affecting others, mostly because they do not need to know each other. (Hein, Böhm, and Krcmar, 2018) A physical example can be electronic devices from manufacturers that allow customers to switch out components and connect to peripherals. The device is built and developed so that these changes can be made without having a negative impact on the device. Later on, one would also have other actors producing components for this device, and in some cases, it is the operating system that becomes the core functionality that the future ecosystem revolves around.

Baldwin and Woodard (2008) uses biology to explain how a modular architecture is crucial for the evolution of a platform by discussing the separation of core functionalities on a cellular level where a combination of stable core processes and complementary variable processes facilitate for evolutionary adaptation. The interdependence of components on a cellular level can be compared to how a platform architecture divides a system into a core and complementary external components, usually through boundary resources provided by the platform owner (Ghazawneh and Henfridsson, 2013). An architecture like this promotes the reuse of core components and reduces the need for rebuilding from scratch when developing a new product.

Cloud Platforms

Over recent years, Cloud Computing has emerged as an industrial trend for enterprises wanting to decrease the cost of resources and still keep the business economy and processes efficient (Zhang, Cheng, and Boutaba, 2010). Cloud computing, or cloud platforms as it is often referred to, is a lot like regular platforms in a way that they provide computing services to end-users, enterprises, and organizations. Computer services can consist of storage, analytical tools, services, network, and databases. Moving over to cloud from traditional solutions may lead to benefits such as cost reduction, global scaling, increased performance and speed, productivity, reliability and security (Microsoft, 2019). They can be public, private, or both, which is often referred to as a hybrid.

- Public clouds are clouds in which service providers offer their services on a cloud to the public (Zhang, Cheng, and Boutaba, 2010). Examples of public clouds are Amazon Web Services and Google Cloud.
- Private clouds are developed and maintained exclusively for one organization. As there are private and sensitive data inside the systems, a private cloud offers greater control of security, reliability, and performance (Zhang, Cheng, and Boutaba, 2010).
- Hybrid clouds are a combination of public and private clouds that address both of the cloud's limitations by providing tighter control and security while they provide on-demand service expansion. This is realized by running part of the infrastructure on a private cloud and the rest on a public cloud (Zhang, Cheng, and Boutaba, 2010). The challenge here is to determine the best split in public and private clouds.

There are different types of cloud computing and cloud services provided. Literature divides them into three core categories: IaaS, PaaS and SaaS (Zhang, Cheng, and Boutaba, 2010).

- Infrastructure as a Service (IaaS) is the provisioning of infrastructure resources (Zhang, Cheng, and Boutaba, 2010) such as computing, network and storage resources needed for base infrastructure (IBM, 2019).

Such a low level provision of resources can be found more cost efficient for businesses by reducing costs of managing their own local infrastructure.

- Platform as a Service (PaaS) provides the necessary platform resources to develop applications and services such as os, middleware and frameworks. In addition to these resources, PaaS resources also include resources from an IaaS. (Mell and Grance, 2011; Zhang, Cheng, and Boutaba, 2010; Microsoft, 2019)

- Software as a Service (SaaS) is the model where the consumer uses the provider's application on a cloud infrastructure (Mell and Grance, 2011). In SaaS, the rest of the service layers are abstracted away (IBM, 2019), meaning that the cloud provider's applications are in focus.

Figure 2.2 illustrates a high level presentation of how these categories can be connected to each other in a end-user serving manner.

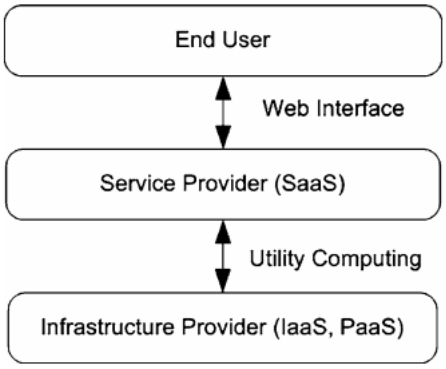


FIGURE 2.2: High level presentation of cloud layer presented by Zhang, Cheng, and Boutaba (2010)

2.2.2 Enterprise Architecture

"Enterprise architecture (EA) is the definition and representation of a high-level view of an enterprise's business processes and IT systems, their interrelationships, and the extent to which these processes and systems are shared by different parts of the enterprise."(Tamm et al., 2011).

Enterprise as a term refers to the whole organization, and will often span multiple organizations (The Open Group, 2009, 3.38). Business processes are sets of tasks or activities which the enterprise follows, to create value for their customers. Mapping the business processes and IT systems within enterprise architecture will in other words show us the organizations business logic and reflect on the integration and standardization of their operating model (Weill and Ross, 2004). EA lays out a long term view of processes, systems and technologies of a company. Focusing on long term needs rather than immediate needs (Weill and Ross, 2004).

An operating model, such as the one in figure 2.3, is used to classify the level integration and standardization of the business processes used to deliver value to the customers. Business integration is used to describe the level of communication between business units and data that is shared. Business Process standardization is used to describe how much alike the different processes are between different business units.

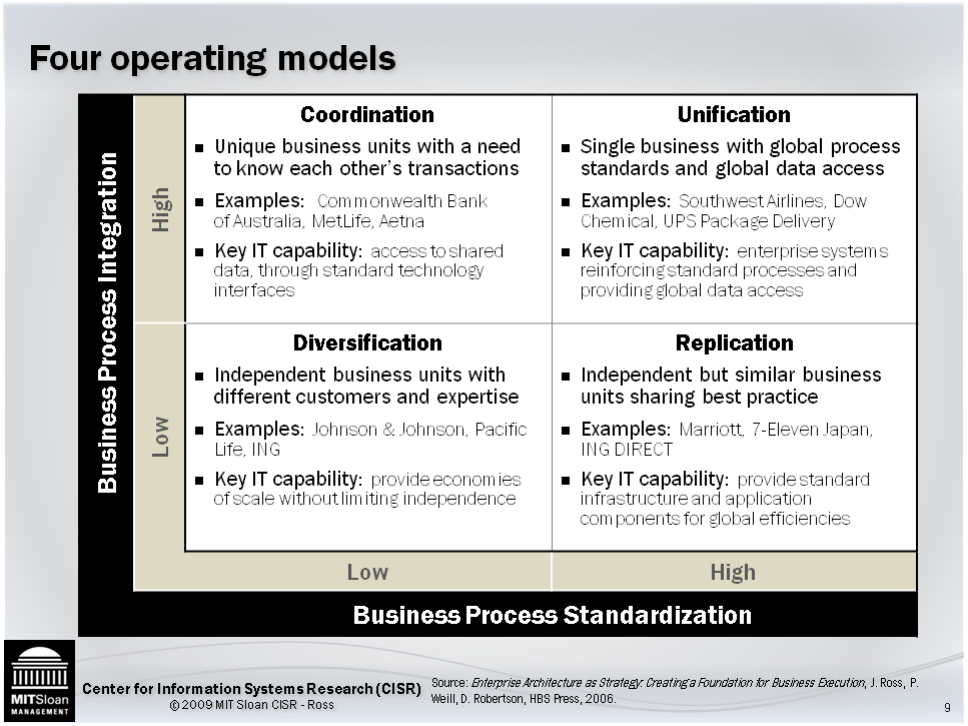


FIGURE 2.3: Operating model weillross

As popular as EA is within enterprise IT, it also meets scepticism as to how it can be called a technical architecture. Bloomberg (2013) mentions that unlike solution architecture and other technical centered architecture frameworks, EA is not something designed before the development of the enterprise solution. He also highlights the argument by saying that "Architecture comes before you build something". EA is a way or tool to improve existing system, but at the same time it is a blueprint of underlying applications and business processes.

2.2.3 Agile Architecture

The Cambridge Dictionary (2020a) describes agility as the ability to move quickly and easily. Agile is an umbrella term, meaning that it is a phrase covering a wide range of concepts.

In software engineering, the term agile software development is frequently used to describe modern best practices when developing software (Bloomberg, 2013) Frameworks such as Scrum and Kanban are among the more popular ones and presents iterative and wasteless work processes within software projects. These are competing with old ways of working with software such as the waterfall model, which requires predefined steps within a project on the cost of not being able to make large changes during the project.

Looking at architectural structures within software and systems, one can draw lines between large monolithic systems and modular systems as old versus modern. Bloomberg (2013) mentions business agility as the ability to respond quickly and efficiently to changes within the business environment. This applies to the IT industry which is prone to constant changes due to new technology and best practices. Taking this into consideration, one should focus on building systems that applies the mindset of business agility, so that the system will be able to support and withstand changes over time, rather than facing challenges related to important changes. By architecting the enterprise in order to achieve business agility, we will get an Agile Architecture (Bloomberg, 2013).

This section will explore the properties that are important to include in an agile architecture, and the most prevalent agile architectures in use. We begin by introducing the terms that are essential when discussing how systems interact: Interoperability and integration.

Integration and Interoperability

Interoperability describes a systems ability to communicate with other systems, without having to go through a lot of steps when requesting and responding. In Health Information Management it is described as "the ability of health information systems to work together within and across organizational boundaries in order to advance the health status of, and the effective

delivery of healthcare for, individuals and communities" (HIMSS, 2020). The Open Group (TOGAF) gives a short definition of interoperability as "the ability to share information and services" (The Open Group, 2009). In the The Open Group, 2009 standard, Interoperability is divided into three main categories:

1. Business interoperability: Defines how Business processes are to be shared.
2. Information Interoperability: Defines how Information is to be shared.
3. Technical Interoperability: Defines how services are to be shared or connected to each other.

Without having agreed on standards shared between systems, you cannot obtain interoperability (Braa and Sahay, 2012).

Integration should not be thought of the same as Interoperability, as people might have the assumptions that integration is about creating a big system and interoperability being its ability to connect with other systems (Braa and Sahay, 2012). Braa and Sahay (2012) define integration as the process of connecting multiple sub systems in a manner that makes them appear as a whole system.

Service Oriented Architecture

Service Oriented Architecture, mostly phrased SOA in the IT-field, is an architectural design methodology that is oriented towards services (Bloomberg, 2013). In Bloomberg (2013) a service in the context of SOA is described as "the business abstraction- that is, a representation of functionality and data presented in a business context. Unlike a platform architecture or enterprise architecture, SOA also works as a goal for systems. Meaning that the more SOA related methodologies one uses, the more service oriented their systems become.
When discussing agile architectures, Bloomberg (2013) introduces SOA as an agile architecture. A core concept of the SOA is loose coupling, which in SOA means that if a Service is changed, one does not have to change the whole system. Components in the systems do not know anything about each other and replacing or changing services does not impact the system in any way. This is important for the architecture because of the system's ability to make changes to different components without affecting other components (Duggan, 2012; Bloomberg, 2013).

The applications and services in a SOA architecture are often connected to the enterprise infrastructure through a middle-ware. This type of middle-ware is often referred to as an Enterprise Service Bus (ESB). The ESB works as a layer of abstraction that enables communication between different services and applications in an enterprise without the differences of implementations becoming obstacles (Oracle, 2020). In figure 2.4, Oracle (2020) illustrates a

generic ESB connected to services and service clients. Middle-ware, such as the one used in the case of the ESB, are system components that work as glue between clients and resources in a distributed enterprise system. Interfaces such as the API are critical for the exchanging of data (Duggan, 2012).

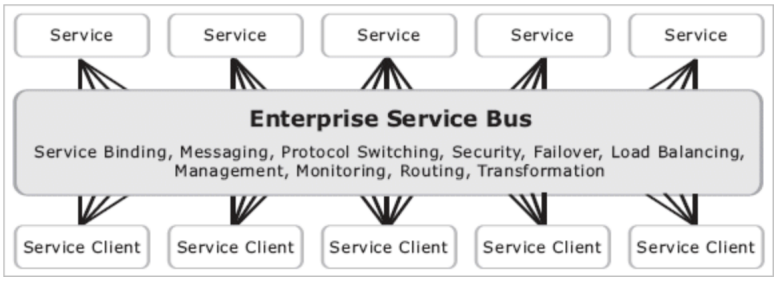


FIGURE 2.4: Enterprise service bus illustrated in the Oracle docs)

REST-based Architectures

Representational State Transfer (REST) refers to an architectural style that revolves around web services. The concept of REST was introduced in Roy Fielding's PhD dissertation in 2000. Bloomberg (2013) discusses the use of REST with SOA to lessen the dependence of Services within the systems in order to move towards an agile architecture.

What makes REST a concept to discuss within agile architecture is the way Fielding and Taylor (2000) describes it as: "REST is a coordinated set of architectural constraints that attempts to minimize latency and network communication while at the same time maximizing the independence and scalability of component implementations.". The definitions of these constraints are shortened and directly quoted from Fielding and Taylor (2000):

- Null style: Null style describes a system in which there are no distinguished boundaries between components.
- Client-Server: Separation of concerns is the principle behind the client-server constraints. By separating the user interface concerns from the data storage concerns, we improve the portability of the user interface across multiple platforms and improve scalability by simplifying the server components.
- Stateless - There are a lot of different constraints that make up REST, but the most prominent one is statelessness. Stateless meaning that during the client-server interaction, the client request needs to contain all necessary information and can not use advantage of stored context on the server side. Session state is therefore not stored on both sides, but only the client site.
- Cache - Cache constraints require that the data within a response to a request be implicitly or explicitly labeled as cacheable or non-cacheable.The advantage of adding cache constraints is that they have the potential

to partially or completely eliminate some interactions, improving efficiency, scalability, and user-perceived performance by reducing the average latency of a series of interactions.

- The central feature that distinguishes the REST architectural style from other network-based styles is its emphasis on a uniform interface between components. Implementations are decoupled from the services they provide, which encourages independent evolvability.
- Layered system - The layered system style allows an architecture to be composed of hierarchical layers by constraining component behavior such that each component cannot "see" beyond the immediate layer with which they are interacting.
- Code-on-demand: REST allows client functionality to be extended by downloading and executing code in the form of applets or scripts.

2.3 Information Security

Information security is the practice of protecting information assets within a system. This practice covers every aspect from low level code to the social aspect of social engineering. Cisco (2019) refers to information security as the processes and tools designed and deployed to protect sensitive business information from modification, disruption, destruction and inspection. Another definition of Information security is from the International Organisation for Standardization (ISO) 27000 standard which defines information security as "the preservation of confidentiality, integrity and availability of information; in addition, other properties such as authenticity, accountability, non-repudiation and reliability can also be involved. " (ISO, 2018). The information we want to protect consist of mostly personal or sensitive data, which is going to be explained later in this section. The literature often has a focus on the triangle of CIA - Confidentiality, Integrity and Availability, which is presented in the standard definition above. Properties such as authenticity, accountability, non-repudiation and reliability has become more important additions to the CIA concepts.

The CIA triangle

CIA is a term that describes the three fundamental principles of information security: Confidentiality, Integrity and Availability.

Confidentiality is the property that information is not made available or disclosed to unauthorized individuals, entities, or processes (ISO, 2018). In other words, sensitive and private data should only be accessed by someone authorized to do so. An example would be that only patients and their doctors should be able to view the patient's medical record data.

Integrity is the property of accuracy and completeness (ISO, 2018). Data that breaks with this property usually is modified data. Unauthorized modification of data happens when attackers compromises the systems integrity, but can also happen when users of the system makes changes in the configurations Harris, 2018.

Availability is the property of being accessible and usable on demand by an authorized entity (ISO, 2018). Users should be able to access information at all times and the organization should be able to recover from down-times easily and fast. How serious the impact of unavailable systems and information is, depends on the type of system and its use. Due to the effect from technical network supplies (such as routers, dns servers, proxy servers and firewalls), software (such as operating systems and applications) and physical factors, the organization needs to have full understanding of the operational environment that surrounds them (Harris, 2018).

In this paper, the principle of authenticity which was mentioned in the ISO (2018) as an addition to the CIA triangle will be introduced.
Authenticity is the property that an entity is what it claims to be (ISO, 2018). To make sure that an entity's claimed characteristics are right, one has to use controls of authentication (ISO, 2018). While these nouns may sound similar, authenticity is about the quality of an entity's characteristics, and authentication is a mechanism to make sure that this is a valid entity.

2.3.1 Identity and Access Management

Harris (2018) highlights that one of the most important cornerstones in information systems is the control of how resources gets accessed. Access control mechanisms of physical, technical and administrative nature has to be implemented. Jøsang (2017) defines access control as means to ensure that access to assets is authorized and restricted based on business and security requirements. Harris (2018) presents access control as security features to control how entities communicate and interact with each other so that they can protect them from unauthorized actions. An organization can introduce such features as for instance locks that can only be accessed by right personnel (physical) or simple log in features in a system (software).

2.3.2 Privacy

Privacy is a topic discussed between several different disciplines and context. Given the section of information security, the definition of privacy will be on that of personal information. Personal information or personal data, is information that can be used to identify a living indiviual (European-Commission, 2020b; Schwartz and Solove, 2014). Which also means that the smallest segment of data can be classified as personal data, if it contains information that can be used to trace a living person. Examples of personal data are: - First name and/or surname. - phone number - identifiable mail adress. - ip adress and data stored by hospitals that usually are unique.
Among personal data, there is data that is considered sensitive. European-Commission (2020b) categorizes following personal data as sensitive:

- Personal data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs
- Trade-union membership
- Genetic data, biometric data processed solely to identify a human being
- Health-related data
- Data concerning a person's sex life or sexual orientation

2.4 Summary

This chapter presented the literature related to the study of the thesis. To first understand a research platform, it is essential to know the concepts regarding the topic. System architecture was introduced and explained to get a broad definition of the term. Literature regarding system architecture was divided into three parts: Platform Architecture, Enterprise Architecture, and Agile Architecture. Even though we have a section discussing the concept of platforms, there was a need to present the platforms' architectural traits. Enterprise architecture works as an architectural methodology that focuses on how businesses want their enterprise to be long-term. Agile architecture, while being a relatively new term, is the main focus of this thesis. Thus we have to include concepts related to this notion. Information security and privacy are introduced at the end of this chapter because of their importance in research platforms concerning sensitive data. In the following chapter, we will create a framework using Bloomberg (2013) to investigate agile architecture within research platforms.

Concepts	Description
Platforms	Digital platforms are Software foundations in which two or more parties interact with each other and need each other. Most commonly seen in mobile platforms like Android or iOS where platform owners provide value to app developers and end-users.
System Architecture	An overview of the components and their interactions in an organization that makes up their software system. Components can be digital, physical and human.
Platform architecture	A platforms architecture is layered and modular. Often consists of a platform core that is controlled by the platform owner, third party such as end-users or platform developers and the boundary resources which is shared by the platform owner that the developers can use to develop complements to the platform.
Enterprise architecture	A high level view of the business processes and IT systems within an enterprise. It shows their interconnections and how the processes and systems are shared by the enterprise (Tamm et al., 2011).
Agile Architecture	Agile architecture is presented by Bloomberg (2013) as an architectural approach that takes the ability to responds to changes into focus. Service Oriented Architecture is often used when describing flexible services within a loosely coupled and changeable architecture.
Information Security	Information Security is the practice of protecting information assets, mainly by preservation of confidentiality, integrity and availability of information (ISO, 2018).
Privacy	The property of keeping personal data unavailable for other entities. The EU has ruled out a new law regarding digital privacy, that has to be followed for systems that use personal data within Europe. Sensitive data should not be traceable, as it is a breach of privacy laws.

TABLE 2.1: Summary of Related Research

Chapter 3

Theoretical Framework

We will develop a theoretical framework based on the chapter "Agile Architecture in Practice" from Bloomberg (2013), to answer the research question. The chapter from Bloomberg (2013) provides characteristics for an architecture that enables change and flexibility. One can use these characteristics as criteria needed for an agile architecture. We will use this framework as a basis for interpreting and understanding our empirical findings.

Thus, this chapter will look into the transition from an integration-centric approach to technology to a composition-centric approach from a technical point of view. The integration-centric view can be looked at as a heterogeneous collection of resources. In contrast, the composition-centric view looks at IT resources as flexible services that can be composed to support and manage flexible services. With these approaches in mind, we will try to create a framework to align agility and platform architecture within a research domain that will be used to answer the research question.

3.1 A framework to align agility and platform architecture within a research domain

Bloomberg (2013) emphasizes that the importance of agile architecture is to move away from traditional IT systems in a way that improves business processes. They have also previously mentioned that Enterprise Architecture is not a form of architecture, but rather an overview of business processes supported by the systems and enterprise. Traditional definitions refer to business processes as "a sequence of activities that leads to a business result" while Bloomberg (2013) defines the term broadly as "anything a business does". In an agile architecture, these processes will be dynamic, which from a technical point is harder to automate the more dynamic they are.

As mentioned in the previous chapter, a platform architecture consists of a software core, complementary applications or services, and the interfaces shared by both of them that allow interoperability. They are imagined to be modular and loosely coupled, decreasing interdependencies within the

system and allowing change and reuse of components without negatively affecting other system components. In a research domain, these services have to be related to the use of research data. The services will have to be able to communicate with the platform core to get the requested data, and this will be through an interface, which allows those two to communicate. Bloomberg (2013) argues that this would aggregate a lot of technology and create a heterogeneous IT environment that increases the complexity over time. Therefore it should be critically important to consider the interdependencies of the services. Businesses should be able to compose services that would implement business processes without worrying about the services' possible negative behavior.

Based on the chapter Bloomberg (2013) presents about agile architecture in practice, we will draw three concepts that will be used in analyzing findings to discuss how one should establish an agile architecture in a research platform. These three distinct concepts are Process, Service, and Loose Coupling.

3.1.1 Process

As mentioned earlier in this chapter, business processes are anything a business does. Often looked like a stream of tasks that are mapped through flowcharts and models. IT business processes standardize all these activities and bring them together to correspond with information technology. Unfortunately, in traditional IT development has been prone to unchangeable business processes in their logic. As a result of this, limitations such as the need to re-designing applications because process logic is fixed and hidden behind in application implementations (Yao, Zhang, and Wang, 2008). Bloomberg (2013) discusses the need for BPM (Business process management) to emphasize flexibility in business process changes in order to obtain an agile architecture.

For these business processes to be executed in an IT system, there is a need to develop Services that support these processes. The composition of these services has to be such that if the processes change over time, they have to follow these changes and adapt dynamically. Rather than focusing on business processes, we should focus on research processes, as it creates more value to the researchers and thus more value to the platform this thesis presents, which means that the notion of processes in this thesis has to be centered around the activities of the researchers.

3.1.2 Service

Furthermore, Bloomberg (2013) presents Service Oriented Architecture (SOA) as an enabler of flexible services that supports business processes because governance and business process management (BPM) will be pulled together to support compositions of the required services if the methodology is used correctly. Organizations today seek to separate the process layer from the underlying applications because they can own and control their processes directly. This way, they can provide their desired flexibility and control. Tensions between control and flexibility arise, as processes might be too formalized (high control and no room for change) or not formalized (no control and high flexibility).

Technically, SOA must enable companies to build Service that represents data available to the business and the functionality of underlying systems. The businesses must be able to configure the requirements even if they change over time. (flexibility and control) (Bloomberg, 2013).

Bloomberg (2013) claims that some environments think of SOA as just a middle-ware based architecture with an ESB. As mentioned earlier, an enterprise service bus (ESB) is a component used within infrastructures of organizations that practice SOA. It is used to connect services to the underlying functionality of the systems. The issue with this kind of thinking, according to Bloomberg (2013), is that businesses focus on solving integration problems with an ESB rather than combining it with the SOA methodology. Legacy application integration and data integration are essential parts of SOA, but the role of the integration should be looked at and discussed. SOA should focus on these integrations as a supportive role, while in an ESB-based integration, they play an essential role (Bloomberg, 2013). In short, these integrations should be a result of Service composition. This also results in Service reusability. One should be able to do SOA without integration.

The composition centric focus is one of the design principles of SOA methodology, in which, according to Bloomberg (2013), service composition should be supported by business services.

3.1.3 Loose Coupling

Flexible solutions might require fewer interdependencies between components and services in a system, which is why loose coupling should be one of the main goals when trying to obtain an agile architecture. As Bloomberg (2013)(p.96) points out;

> "Nevertheless, if you're able to get the architecture right, then you'll be able to shift the focus of integration from the tightly coupled, technology-centric world of connecting systems to the loosely coupled, business-centric world of composing Services, and you'll be well on your way to an Agile Architecture."

Loosely coupled systems tend to be viewed as systems with independent components that do not act responsively. In contrast, tightly couples are viewed as systems with responsive components that are not independent. Following, loose coupling is often portrayed as an endpoint on a scale from loose to tight coupling (Orton, Douglas, and Weick, 1990). Interdependence is often used when talking about coupling in systems and describe mutually reliant parts. Loose and tight coupling can then be looked at as the degree of interdependence in a system. The notion of loose coupling first came to be in the spotlight after Parnas (1972) discussed dividing a system into modules. As modularity is an effect of loose coupling, Parnas (1972) would come to give one of the first technical views on the loose coupling.

Bygstad (2017) proposes the notion of loose coupling with a focus on integration and proposes three design principles that should be taken into account when looking into the degree of loose coupling in a system. Bygstad (2017) discusses that systems should be loosely coupled (1) technically, (2) in terms of standards, and (3) in terms of organizations. Technically, a system should be loosely coupled by allowing the lightweight technologies to support work processes before they are integrated. Standards should be means, not aims, and therefore there should be less focus on trying to standardize all processes within a system. Organizational, two solutions are proposed: Heavyweight vendors providing their platform as an innovative place for third-party developers. Or thin calls from heavyweight to lightweight technologies within an organization.

Following this chapter, we will have three themes to focus on when collecting and analyzing data for agility within an architecture: Processes, Loose Coupling, and Services.

Looking at these concepts in relation to platform architecture, we have to classify what is required to make a platform architecture agile. Using the integration-centric versus the composition-centric view from Bloomberg (2013), we need to look into how the components interact with each other through the layered modular architecture of a platform. Discussing the tension between flexibility and control of the organization's business processes and how governance and architecture affect the agility of the platform.

3.2 Summary

To summarize the content of this chapter, we derived a framework based on Bloomberg (2013) discussion regarding agile architecture in practice. In this framework, three concepts were focused on as important when investigating the agility of a system. In our case, a digital platform system. Beginning with processes, we need to see and map the different processes within the organization. All these processes that are mapped need to have services that support them. It is then necessary to discuss how they are developed and composed. Lastly, it is crucial to see how the components or services are interconnected within the system. A tightly coupled system might not be as prone to continuous changes as a loosely coupled system is. The following chapter will present the research methodology used in this thesis.

Concept	Description
Processes	By Processes, we focus on research processes in this thesis, as the topic of discussion is related to the research domain. Within Agile Architecture, enterprises should focus on having BPM that supports flexibility of processes. In a research domain this will mean that the system also needs to emphasise the importance of activities of researchers.
Services	Moving away from integration centered SOA to a more composition driven SOA we can enable flexible services that supports business processes even if they are prone to changes. Another concept to look at here is the re usability of services, which can also support changing processes.
Loose Coupling	Within an agile architecture, one might require less inter dependencies between components to offer flexible solutions. Parnas (1972) introduces loose coupling as separating a system into modules. Modularity is an effect of loose coupling, which is why we need to look into the modularity of the system we analyse to get a grasp of the degree of loose coupling. Bygstad (2017) discusses that loose coupling in systems should be done technically, in terms of standards and in terms of organisations.

TABLE 3.1: Summary of Scope

Chapter 4

Methodology

This chapter presents the methodology and research design used in this thesis. First we present the chosen research method and design, and why they were chosen, followed by an in-depth look at the research design. We will discuss ethical consideration in research, at the end.

4.1 Research Method

As this research paper explores the complex picture of a research platform, the research method chosen is qualitative research (Creswell, 2007). Various sources of data gathered by the researchers themselves, such as interviews, observations and documents, are a part of qualitative research (Creswell, 2007, p.38). This thesis also relates to some of Creswell (2007) presented characteristics, such as a holistic account, theoretical lens, interpretive inquiry, and researcher as a key data collector.

Although qualitative data is what we aim to obtain in this research project, we might obtain useful quantitative data. Quantitative research, unlike qualitative, is used in empirical studies by collecting, analyzing, and displaying data in a numerical way rather than narrative (Given, 2008).

4.2 Research Design

We seek to gather qualitative data through a case study, which is a research design strategy that Yin (2014) suggests when studying an issue within a context.
«Whatever the field of interest, the distinctive need for case study research arises out of the desire to understand complex social phenomena. In brief, a case study allows investigators to focus on a "case" and retain a holistic and real-world perspective—such as in studying [...] organizational and managerial processes [...] and the maturation of industries.» (Yin, 2014, p.56). In this thesis, the case study will be used to understand TSD as a unique research platform within the context of agile architecture.

Yin (2014) defines three types of case studies for research: Explanatory-, Descriptive- and exploratory case studies. This case study seeks to be an explanatory case study, and meets the three conditions Yin (2014)) presents:

- Seek to explain why a "how" and "why" a phenomenon occurs.
- Seek to explain a contemporary phenomenon
- The researcher must have no control over the phenomenon.

We seek to understand the term agility in system architecture and how TSD, as a research platform, aims to be agile. As a researcher with no ties to the TSD group, we have no control over how the system is designed.

4.3 Data Collection

4.3.1 Interview

Interviews are often used in empirical software engineering research and are often used to collect data about phenomena that we can not observe through quantitative research (Hove and Anda, 2005). According to Yin (2014), interviews are one of the most important methods in gathering information in a case study. Thus they are commonly found in case studies. They are either long (Prolonged case study interviews) that last 90 minutes or more, or short (Shorter case study interviews) that last up to one hour.
Interviews can be categorized into three types, based on the fluidity of the conversations (Yin, 2014):

Interview Type	Description
Structured interview	Questions in a structured interview are predetermined and the interviewer follows an interview guide with a set of questions, from start to end.
Semi-Structured interview	The interviewer follows a set of questions, but keeps the conversation fluid by asking questions made up during the interview.
Unstructured interview	Interview is fluid from the beginning to the end and none of the questions are pre-written. This may resemble more of a conversation, rather than a stream of queries (Yin, 2013).

TABLE 4.1: Three main types of interviews used in Case Studies

We will be working with USIT, University Center for Information Technology, throughout most of this project. So we will need to conduct interviews to get new data and information about the TSD project during the research. Interviews will be semi-structured, so the participants have the freedom to offer their thoughts and new input to the topic of study (Galletta and Cross, 2013, p. 3). This also means that we will obtain information that varies from each person and their role in the organization. It is important to remember to ask non-threatening questions and, at the same time, get the answers one seeks (Yin, 2014). We interviewed seven people during this case study. First, the interviews were limited to people with direct connections to USIT. However, as the case study moved on, we had to interview new subjects. These are researchers that use the platform to do their respective jobs. We interviewed these researchers to get a better picture of what their role and needs in the platform ecosystem were.

Role	Number of Interviews conducted
Assistant Director	3
Section Manager (Developer)	2
Head of Web Development	1
Researcher (Bioinformatics)	1
Researcher (Economics)	1
Researcher (Neuroscience)	1

TABLE 4.2: Interview Subjects

4.3.2 Documents

Examining documents can help understand the requirements set for developing a new architecture, as some principles and rules need to be followed. In addition, the existing documents about the infrastructure can help us get a better insight into the different components that make the architecture. Questions for the interviews can be created by using the information found in the documentation (Yin, 2008).

Literature Review

The use of relevant literature helps strengthen the theory around the research topic and backing up arguments directed towards the research question and the project.

Table 4.4 presents the three types of documents used to gather empirical evidence in this thesis.

Documentation type	Description
White Paper	How the project came together, and a summary of the contents and components.
Presentation	Documents and Presentation slides, used to present TSD to users and other third parties. Works as a light description of the project.
Technical documents	An overview of the systems infrastructure, its components and interconnections. Documentation of the API design TSD use, explaining how this is used by external users.

TABLE 4.3: Types of Documentation gathered in this Research

In this section, we discussed the importance of three data sources in a case study. Interviews and Documents are among the sources Yin (2014) discusses, which we chose for our case study.

Method	How	Why
Interview	Interviews with individuals that are related to the subject of study. Semi-structured interviews following an interview guide.	Gaining insights through participants that know the system in-depth and users of the system.
Documents	Obtaining technical documentation, presentations and whitepaper that would explain the subject in detail.	Gain insight in the specifics of the subject and the concepts revolving around it. Documentation sent by the organisation is solid evidence.

TABLE 4.4: Data Colletion Methods

4.4 Data Analysis

In this section, we discuss how the collected data is going to be analyzed. The analysis was conducted in 5 steps. First, we went through and mapped key events in the evolution of TSD, from when it was established to the research platform it has become today. By doing this, we could identify challenges that have been present in the development of TSD over the years.

Secondly, we collected data and identified the various activities and issues related to building TSD. We carefully read through all the documents sent by the TSD team. The documents covered topics from low-level technical detail to higher-level details related to projects and risk assessments. Given the nature of the documentation regarding a secure platform, we first iterated through what would be appropriate to introduce in the thesis. We then mapped pages relevant to the research questions.

Data collected through interviews are first recorded and later transcribed into a document with all written statements verbatim. Transcribing interviews is a time-consuming task; an hour of audio can take around eight hours to transcribe but strengthens the integrity of the content by being true to what has been said (Hove and Anda, 2005). We used the different sources of evidence with each other; for instance, earlier interviews and documentation were used to tailor the interview guides to fit each interviewee and the information we had to obtain. For instance, we tailored each interview-guides to fit the interviewees by using earlier interviews and documentation.

For the transcription, we used regular recording software to record and play the audio from the interview. While listening through every interview several times, we wrote down each word on a computer.

Data collection and key events are results of the dynamics between them. By this, we mean that data collection did not only happen after going through key events. The first step was a result of our first data collection, but in the second step, we saw that TSD would fit the agile description in the way it worked. It was supposed to be flexible, create applications and services continuously for research projects, and scale with a large user base. From the data collection, we noticed that the agile architecture became central in TSD. Following this, through interactions between gathered data and attained theory, we had to establish a framework that focused on the central concept in agile architecture.

Thirdly, we gradually created a framework for the fundamentals of agile architecture. The data collected showed us that TSD is trying to be agile and has had challenges related to agility. Not just agile in the form of software development, but as a whole system. We used concepts from Bloomberg (2013) with a focus on agile architecture in practice. Concepts such as processes, services, and modularity are central in agile architectures that are supposed to respond quickly to changes, dynamic, easily established, service-oriented, and secure.

Fourthly, we analyzed the data in light of agile architecture, using the framework we built in chapter 3. By following the framework, we highlighted findings that matched with the three topics presented. While the literature from Bloomberg (2013) does focus on systems different from our study, we managed to find connections between the TSD platform and concepts of the framework. Besides, we presented the notion of security in empirical evidence due to the importance of sensitive data in TSD. Using the empirical evidence alongside the framework, we analyzed the research platform's agility and the implications the strict security has.

Finally, we discussed and theorized the findings in relation to the relevant literature. We discussed agile architecture within a research platform such as TSD, and how such an architecture could be established.

Activity	Result
1. Key events and central events in the development of TSD	Chapter 5 and Chapter 6
2. Collecting data and identifying the various activities and issues related to building TSD.	Chapter 6
3. Gradually creating a framework for the fundamentals of an agile architecture.	Chapter 3
4. Analysing the empirical data using the theoretical framework.	Chapter 6
5. Discussing and theorizing findings in relation to the literature.	Chapter 7

TABLE 4.5: Data analysis

4.5 Ethical Consideration

This section will discuss the ethical aspect of collecting data through qualitative means such as interviews and documentations used in conducting the case study.

Writing about TSD

Writing a study about TSD had few challenges when it came to collecting data. The organization was excited that a student wanted to write a thesis about their system. The willingness also came from our project in collaboration with the organization.

Participation in Interviews

Obtaining information through interviews require actual interview participants. These participants need to know that they are safe in terms of the interview contents and provide them information on what the project will contain and its aim. Before conducting the interviews, the participants were let known what kind of questions were to be asked and the thesis's goal.
It is also important that the questions and topics presented in the interviews are not violating the participant's integrity. They should be on the point and not out of line, meaning that the interviewer should not ask questions that will discriminate against the organization, system, or the participant. One has to distinguish between journalistic digging and a case study interview.

Post interview

At the end of each interview, participants are asked if they want to have a final draft of the transcription. This helps build trust between the two parties and assures the participant that the research paper will not contain any misinformation or falsified testimonies.

Written Consent and Data Protection Service

To start with, each of the participants consented to the study per email. The email threads of each participant contain a description of the project. Later reflection on the topic of consent brought up the issue with clarification over email. Moreover, letters of participation with project description and required field for signature should be sent to all participants that are not part of the project between us and the organisation.
Data protection services, shortened "NSD" in Norwegian, is a national archive for research data. In our case, we have to notify the NSD about our research because the University of Oslo has an agreement with NSD. The thesis has NSD acceptance by being a part of a project named "Digital University" and not presenting sensitive data. The NSD form can be found in the appendix.

Confidentiality and integrity of documentation

As there has been sent some documentation from the TSD group, they still have to be updated on which documents we will use in the thesis. Besides, the organization must be assured that no one will tamper with the information within the documents.

Constructing Validity

Constructing validity is one of four tests introduced by Yin (2013) "to test the quality of research designs, and involves identifying correct operational measures for the concept being studied". The measures used by the researcher have to be related to the object in the study. In addition to multiple sources of information, Construct validity can be tested by a chain of evidence or having the draft case study reviewed by key informants. The three other tests are Internal Validity, External Validity, and Reliability.

Internal Validity

Often used in exploratory case studies, Internal validity seeks to explain the relationships and causes of the different variables in the study. There are five analytical tactics for strengthening internal validity: Pattern matching, explanation building, addressing rival explanations, and using logic models (Yin, 2014).

External Validity

External validity involves taking the conclusions of a study, comparing them to the outside context, and seeing how generalized the findings are (Yin, 2014). The use of research questions often strengthens it.

Reliability

To make the case study reliable, for example, if another researcher were to conduct the same study, they should arrive at the same findings and conclusion. The goal is to minimize errors and biases in a study (Yin, 2014).

The following chapter will present the empirical evidence gathered using the methods presented in this chapter.

Chapter 5

Case

In this chapter, we will present the case of TSD and the organizations revolving around it.

5.1 Background

5.1.1 University of Oslo

The University of Oslo, founded in 1811, is the first and oldest university in Norway. As a pillar of research and higher education in Norway, the University of Oslo hosts around 28000 students and 7000 employees. It is divided in 8 faculties and has 10 ten research center, both specialized and interdisciplinary (UiO, 2020). As of 2020, UiO is the only Norwegian university listed as a top 200 institution on the times higher rankings. This is a world renowned ranking that "[...] is based on 13 carefully calibrated performance indicators that measure an institution's performance across teaching, research, knowledge transfer and international outlook."(Times, 2020). Making research a big part of the values that UiO presents.

5.1.2 USIT

USIT, The University Center for Information Technology is a part of the University of Oslo. It works as the IT-department, and has three areas of commitment: research, education and applied knowledge as presented underneath(USIT, 2018).

- **Research**: USIT provides services and resources in the form of software, computational resources, storage services, access to data collections and advanced support for university research.
- **Education**: USIT provides services and solutions that contribute to the development of teaching and learning quality, students' digital literacy, availability of services and best practice development in the education business. USIT is an advisory expert group for the higher education sector in cases relating to IT in learning.

- **Applied Sciences**: USIT provides services and solutions that simplify access to information, knowledge and data collections that the University has at its disposal, and supports development of the researchers' presentations of important findings. USIT develops and provides tools for the retrieval of cultural and natural history research data. In addition USIT is an important partner in the digitalization of museum collections.

5.2 TSD

Research has changed over the time, whether it be moving research data from paper to computer systems or creating analytical tools that helps researchers use data in a more efficient matter. In this digital shift/transformation, the research groups belonging to the University of Oslo found a need for storage and processing research data. An answer to this was USIT's TSD. As the University of Oslo describes it, TSD is a"platform for collecting, storing, analyzing and sharing sensitive data in compliance with the Norwegian privacy regulation"(UiO, 2017). While mainly being developed and operated by the University of Oslo, TSD is also a part of the national infrastructure for handling and storage of scientific data, NorStore (UiO, 2017).

Information flow

Because a majority of the data used in TSD is sensitive, there are strict protocols to how this information flows in the system. For instance, only administrators of projects are allowed to export data, while the rest of the participants may import data to the TSD. All data that is going to be sent and processed within the platform has to be encrypted, due to the nature of the sensitive data.

Data Storing

Collection of data within TSD is done by using USITs own developed Nettskjema. This software is unique to TSD and is the central component used for everything revolving the collection of data. Nettskjema has a design that resembles net forms, but can also be used with the Nettskjema API.

With the increasing focus on privacy and security, one has to use bankID or MinID provided by the government, in some of the projects. These are unique identificators given to all citizens. In addition, TSD introduces Digital Consent Forms, that through the GDPR regulations allows consenters to view the data they have shared.

TSD events

USIT began developing the pilot for TSD during the years 2008-2011, but found the non-elastic solution to contain numerous flaws. As they wanted to have a solution that would meet the needs of an evolving userbase they started to work on a new version of TSD. The development of the current TSD project began in early 2010, but failed halfway. However, the project got a new start in 2012 when the current director entered as a project manager. The new version of TSD was launched in 2014 as a platform solution, and is still under continuous development. In 2018 it was decided to begin utilizing REST-api to better the architecture in a way that made the architecture of the system more flexible. By undergoing this change they were able

to offer the platform's services more efficiently. Self-service and digital consent were two important features that came in 2019 as more user oriented services. The most recent change was the change from their current IAM named Cerebrum to their own solution that would better accommodate the various research groups in the platform, which we will present in the next chapter. The following chapter also presents our empirical findings, through the chose methodology of this thesis.

Chapter 6

Findings

This study aims to explore the fundamental processes and services of the TSD architecture. We will use empirical findings to present a timeline of how, why, and when the TSD development group had to establish a new architecture to support researchers' processes.

First, the University Center of Information Technology began to see a demand for a service in which researchers of the University could store and process their research data. As scientific data can be private, and some even contain sensitive personal information, they need to be kept safe. The first pilot of TSD was developed from 2008-2011 but halted until Gard Thomassen joined the organization and helped develop the system from 2011-2014, which is still under continuous development today.

Researchers at the UiO and other public research institutions have used the system, and TSD has passed a thousand projects by now. When USIT first began developing the TSD platform, they did not foresee such a demand for the services. The project was estimated to scale to roughly 50 projects and 500 users, and the user base exceeded this estimate. A growing user base leads to a requirement and need for the system to scale with users and adapt to the use of different research environments. In modern times of digital data, researchers want to store and work on sensitive data, on safe facilities, which TSD has solved by allowing users to execute research tasks on their servers, providing a platform for many research environments. With a growing user base of research partners, the platform also needs to develop flexible and customizable services and features.

Through the findings chapter, this thesis will spotlight how the organization builds and develops the system and how this impacts the system's flexibility. The main focus is to investigate how the platform manages to be agile within a secure research environment. Using the literature of Bloomberg (2013), we created a framework in chapter 3 that looks into three concepts that are important to investigate to discuss the agility of a system. According to the framework, we will divide the empirical evidence into three parts based on processes, loose coupling, and services.

6.1 Processes at TSD

Following the concepts of the framework derived in chapter 3, we will need to map some of the essential core processes for most researchers using TSD. Prominent processes were described by the TSD director. We list these up in table 6.1 and explain them underneath:

P1 and P2: Identifying and defining research projects and Applying for resources

Researchers identify essential research projects but need help to clarify the data collection process and an application for research funding from Forskningsrådet. Forskningsrådet is the research council of Norway and decides whether a research project is eligible for funding or not.

The TSD group has a supporting role in defining the different research projects and enabling data collection from their respective fields. UiO certifies TSD as a mandatory platform for storing sensitive data and is recommended for researchers as a safe facility to consider when doing research.

P3 and P4: Data collection and facilitating data collection

When using a research platform, researchers find the need to use tools to collect data for their projects. Data collection is the process of collecting and measuring data from research subjects in an academic manner.

> "- [...] Research assistants that were included in the data gathering, uploads the results in Nettskjema. [...] " - Neuroscience Researcher

Data collection is a central activity in a research project that TSD supports by providing researchers with digital tools. As we saw in the interviews earlier, the researchers all began their work by collecting data and preparing them.

Facilitating the data collection is a necessary complement, as not all data can be of the same type. Neither can they be collected in the same way. Different research projects collect different data in their way, and this must be supported by allowing them to configure data collection tools or develop their own.

P5: Importing data

Researchers might seek to work on data that they already have obtained on their own devices and system. This was mentioned in the interviews with the researchers:

> "[...] Here we also have the first data reduction. Then the lighter batch of data needs to be imported via TSD.[...]" - Genome researcher

> "[...] I log on, zip the files and import them. [...] " - Economic researcher

In these cases, TSD supported the import of data in different formats for researchers. This is supported by making automatized solutions that enable TSD to exchange information with other systems.

P6: Analysing and Processing

Data from research has to be worked on, whether it is by comparing them, calculating, inspecting, reducing, or modeling. This falls under the process of analyzing and requires support in the form of processing.

> "[...]From there we use both self developed and TSD-developed software to do the second data reduction. By data reduction we mean analysing and processing the data we have: For instance, Identifying and comparing dna and tumors. Or applying knowledge in the database to the gathered data. [...]" - Genome researcher.

> "- An example of how we use TSD is: Magnetic Resonance (MR) data is collected in a disc and inserted into a machine. The data is uploaded and through a script, pushed into a database. [...] Processing the data in colossus, get results, analyse and export. Often using applications we develop ourselves." - Neuroscience Researcher

If we look at the genome researcher and the neuroscientist cases, they might not have been able to work on their data with their local resources. TSD provides the resources and infrastructure for these research groups to support analyzing and processing data. Using these resources, the research groups can also develop applications and services on their own.

P7 and P8: Security and Data Storage

When researchers work with sensitive data, there are strict regulations that decide how these should be kept. Data that can be traced to individuals are considered sensitive and should be kept only inside the research group and visible only for authorized personnel. This requires that the platform offers sufficient security and a place to store these data. TSD was created to satisfy researchers' need for a regulated, secure research platform for storing and processing sensitive data.

P9: Publication of data

The publication of data is one of the last processes within research projects. Researchers need to be able to show the results of their work. Depending on the data, one needs to either encrypt the results or publish them as they are. For sensitive data, this means that everything published needs to be non-traceable.

> "[...] Lastly, we need to export data anonymously for further use."
> - Genome researcher.

Researchers from the domain of bioinformatics tend to use the research platform to analyze and process large chunks of data related to DNA sequences and have to be done in a safe environment such as TSD to avoid sensitive information to fall into the wrong hands.
Looking towards the field of economics, we presented how a less processing heavy domain uses the platform. Bioinformatics and Economics are two different fields of study, but the researchers still had some similarities in how they used the TSD platform. These three interviews are based on how researchers from different fields use TSD and give us insight into what the users expect of the platform. We see similarities for mainly analyzing, processing, and securing data storing. Two of them also said that developing services themselves, either by reusing other services or creating from scratch, for using the research data was necessary.

Processes	Description
Identifying and defining research project	Researchers want to define their own projects, and decide what is happening within their projects. An example can be the deciding the roles of different members.
Applying for resources	Being able to request more resources or services. I.E more storage space, processing power or tailored services.
Data collection	Researchers need to be able to collect data.
Facilitate data collection, including testing	Collecting data in a manner that suits the research projects. Facilitating data collection so that researchers can collect the data they want in the way they want. Also being able to test this.
Importing data	Importing data from own systems and devices in a secure manner.
Analysing and Processing	Working on data (files). Researchers might lack the resources to be able to work on them on their own local facilities, so they need to be able to do it on the platform. Requires that the organization provides effective infrastructure that can handle analysing and processing of data.
Security	Data must be kept secure, especially since most of the data is sensitive.
Data storage	Providing a facility to store data. For data in research, the data stored must be kept secure.
Publication of data/ Representation	Results of the research done in projects have to be presentable. For instance through exporting to documents or visualisation in applications.

TABLE 6.1: List of important processes in TSD

From the interviews, we received information about the processes that are present when establishing a project and configuring the data gathering tool. These describe a researcher's processes from establishing a project to using services to working with data:

To begin with, researchers have to apply to USIT in order to create a research project within the platform domain (P1). Project members will be sent credentials if they are not already registered in the systems. This used to be sent manually through post, but now the process has been automated. A project will have one project leader/manager, which will have extra privileges such as exporting data.

From here they will be able to start configuring Nettskjema, the service used for collecting data within TSD (P2). After configuring the service, the researchers of the project can start gathering and importing data for the project (P3).

> "You create a project in TSD, you connect your Nettskjema to the tsd project, then start collecting data. " - Head of web development

After an interview with the vice director, we mapped some of the processes that make up TSD.

The data imported to TSD through Nettskjema have to be stored securely inside the system (P4). Authorized project members or project participants are the ones who should be able to view the data. As a result of the emergence of GDPR, the system needs to be able to share data related to participants that participate in gathering data for the project.

Additionally the data stored are processed and used both in research or to develop new applications on top of the services (P5). Through high processing power, researchers can process heavy data which they can not on their own computers. They can also use services that are developed by themselves or the TSD team on top of the platform, to present, visualize or use the data in the project.

P1 and P2 are formal processes that are mostly affected by the organisation, while P3 to P5 are affected by both the organisation and technology.

6.2 Services at TSD

To fully support researchers' processes in TSD, the system needs to design and develop services that focus on supporting the processes. The services also need to be flexible in a manner that allows for change if the processes change. Therefore we look into the composition of services, the processes they support, and the methodology used when developing them.

In the previous section, the last interview with the vice director of the USITs TSD team was presented. Here he presented the processes that TSD facilitates and also the core Services that support them. Presented in the table 6.2, we see the services that support each of the processes.

Processes	Services
Identifying and defining research project	Self service : User and project administration. Granular access.
Facilitate data collection, including testing	Creation of application by re-using Nettskjema.
Importing data	Using api or tsd.uio.no to import data to your TSD research project.
Data collection	Applications. Nettskjema.
Analysing and Processing	Heavy processing: MPI (message passing interface). GPU. Linux software that can execute on VM. Mat Lab. R. SAS.
Security	Two factor authentication login. Security by Design. Updating security assessment. ISO.
Data storage	Local storage. Databases.
Publication of data/ Representation	Encryption of data. Non tracable data. Self made applications by researchers or tailored applications by USIT.

TABLE 6.2: List of important processes and services that support them.

In the beginning, when someone became a user of the system, they would receive credentials physically in the post. This was not a reliable nor secure way to do this, and as such, the way they handled the process had to be in line with the modern registration standards. Today, the nationwide service minID is used to verify and log in users in TSD. Administrating the users and projects have been automatized to the degree that everything is self-service,

which means that through a user interface, one can handle internal changes themselves.

> "That is the whole design of nettskjema and tsd; self service. Everything we focus on is that people should be able to use this themselves without getting help from anyone. Which also is a difference from other research platforms. They always need support. "

Self-service has become an increasingly important concept within TSD, especially when configuring and developing own applications and services on top of the available resources. Nettskjema is a service that promotes this notion.

Nettskjema - The backbone to services in TSD

One of the most critical and vital processes a research platform like this has is gathering data. It has to be done in a way that is secure, efficient, and configurable. Nettskjema is the core component that collects data from researchers and organizations. Several of the Interviewees have pointed out that Nettskjema, the form that collects data and information, is one of the essential components TSD has.

> "Right now nettskjema is a very important service. In the beginning it was a small bonus and now it is one of the most important part of the TSD ecosystem. The collecting of data outside of the web. It turns out that no one was able to do that in a way that we do. Many researchers come to us just to be able to collect data in a secure way and be able to watch them in TSD." - Head of Web Development USIT

Nettskjema allows importing of data through a standard user interface or through the use of its API. When it comes to developing applications for collecting data, Nettskjema works as a backend. Testing environments for testing the different "Nettskjema" are provided within the platform.

Not every user necessarily uses Nettskjema to gather data for their research projects in TSD. Some of the interviewees from the last section imported their already gathered data to their research project. This can be done through the API provided in https://api.tsd.usit.no or through a user interface on https://data.tsd.usit.no, shown in figures 6.2 and 6.3. The user interface requires a two-factor authentication login, with the user's regular password and a one-time password.

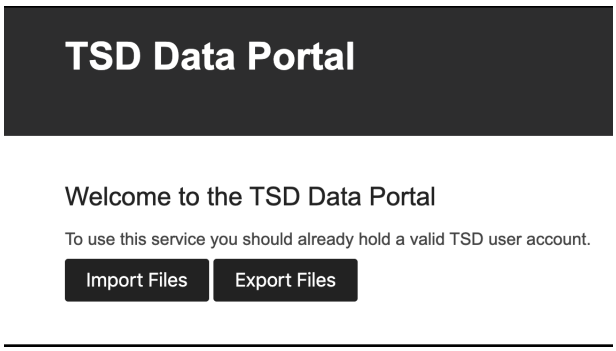


FIGURE 6.1: Import and Export UI

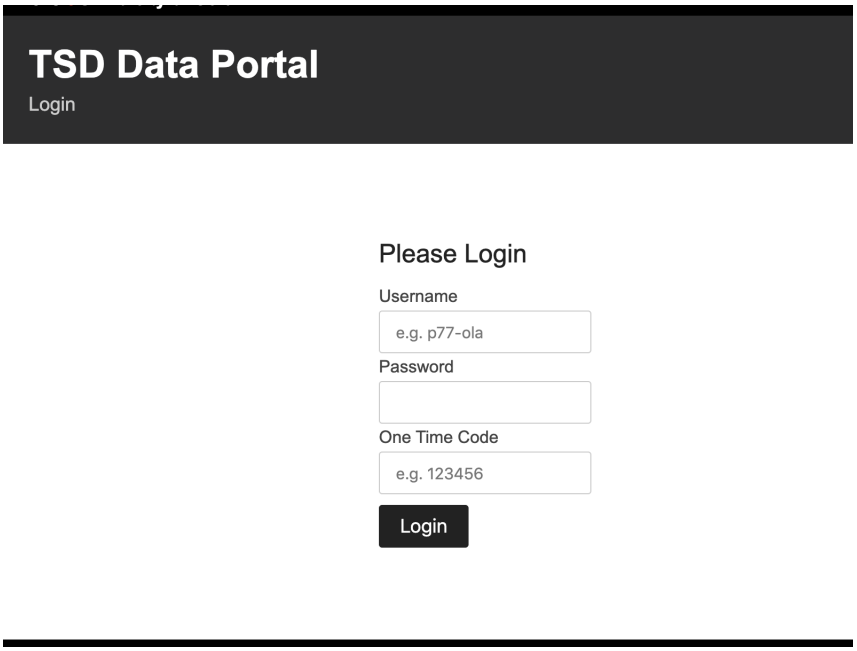


FIGURE 6.2: Login for data import

Security in the research Platform

As research platforms such as the subject in the study focus heavily on security, it becomes essential to mention the notion of security. Security can bring more challenges to how the platform is designed and structured. It is important for eInfrastructures to focus on sensitive data to have control over its data and components.

In TSD, importing data is an easier task than exporting. There are different reasons for this: importing has the issue of containment of malware, while exporting requires knowledge of what kind of data is exported, who sees the data, and assurance that the export is authorized. A solution in 2012 was to limit the functionality of "copy and paste" for data supposed to be exported, and the use of the functionality required two-factor authentication. Figure 6.3 depicts the use of two-factor authentication when logging into TSD.

Sensitive data

"We almost only have sensitive data in TSD"

TSD as a platform is intended to treat, store, gather and share sensitive data. In research these also include business secrets and privacy secrets. Per today, 2-3 petabyte of sensitive data is stored, making it one of the largest collections of personal sensitive data in Norway. At the University of Oslo there is a document called LSIS (Ledelsessystem for informasjonssikkerhet) that contains definitions of data categories TSD uses:

Code	Classification
Green	Open Data
Yellow	Limited Data
Red	Confidential
Black	Strictly Confidential

TABLE 6.3: Table of data categories used in TSD

Red and Black data are the two data categories of greatest importance in TSD. Red data contains data that the University of Oslo is obliged to protect by law, agreements, and regulations. Black data contains data such as red type but requires even more protection as the consequences of tampering are higher. Challenges they frequently meet when developing within the system are related to these heavily regulated data types.

As much as security is the one of the most important concepts that TSD promotes to the users, it can also become a challenge. A challenge in the form of hindrance and complexity when developing new services that needs to be a core theme of discussion.

Data storage is a process closely related to the security of information within TSD. Researchers data has to be kept in a safe environment, and TSD supports this by providing local storage at their facilities, protected by their security mechanisms and architecture. Much of the data stored are in related to the import, collection and processing services of TSD.

"Never store data outside of TSD. The TSD handles security. "

For the analysing and processing of data, TSD offers heavy machinery in their infrastructure with services that researchers can use. Examples here are the services like their Dragen built to utilize their colossus machines. Dragen is a node on Colossus with a programmable processor. This is used for calculations in bioinformatics research, as we heard from one of the interviewees earlier. Research groups can also develop their own solutions that uses this processing power, such as the tools that the genome researchers have developed for their own use.

To present the results of from the reseach projects, the system should first handle encryption of data. This way it can not be traced to specific individuals. Then it can be shared either through applications made by researchers, or tailored applications developed by the TSD team.

FIGURE 6.3: Services TSD Supports

When developing services and components, the TSD group has emphasized that the Service Oriented Architecture (SOA) methodology is their guiding principle. The degree of service orientation can be seen in how the system is designed to allow better communication between services and the system.

6.3 A loosely coupled and modular system for TSD

> "In the beginning it (TSD) was like an non-elastic cloud where we tailored service packages. A windows machine, Linux machine and software." - Director

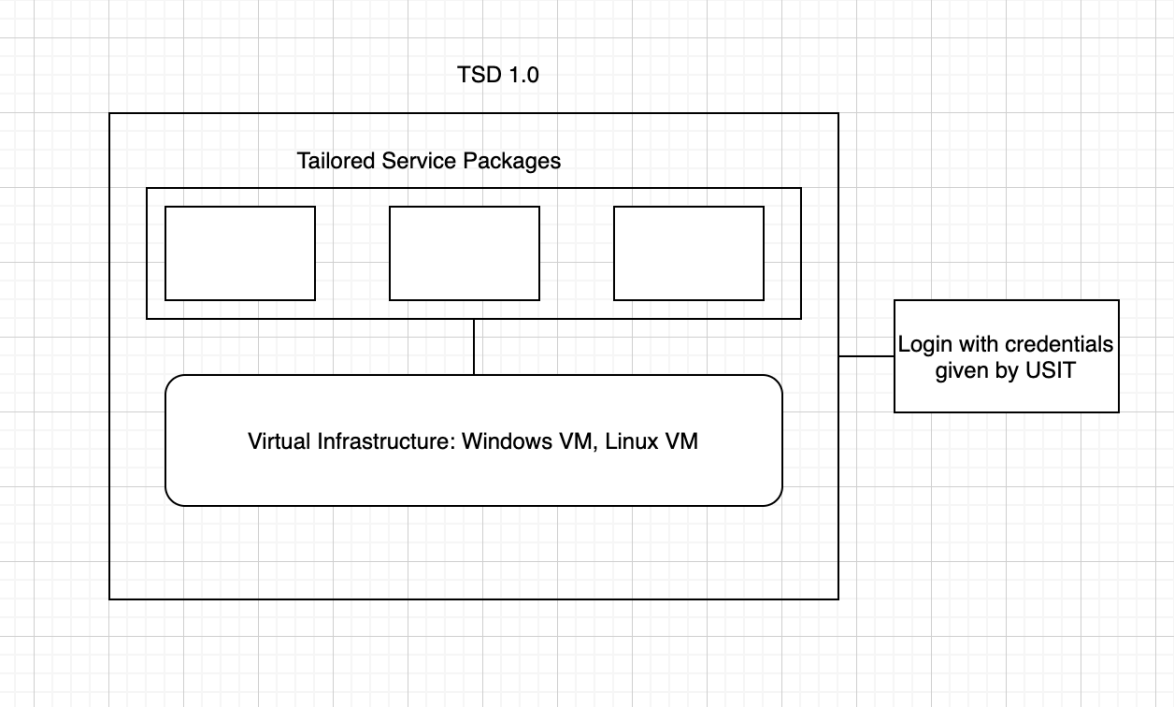


FIGURE 6.4: TSD 1.0 Model

The description of TSD as a non-elastic platform following a confirmation from the director, implied that USIT had developed the system as a monolithic system. When the demand from users and interested research groups surpassed what they had anticipated, the TSD group had to find a way to scale the system. User needs could not be met if they had to develop each single services that each user needed. There had to be a way for delivering own services in a platform-like manner. In addition to scalability, the challenge of maintainability had to be met. These two challenges were the main catalysts for the decision of shifting to a "platform-cloud like solution". One of the developers explained some of the new requirements he met:

> "There was a list [...] The first thing is that it should be two-way authentication. There was to be strong separation between the data. There was to be heavy processing power. There was to be windows machines, linux machines. One should be able to store data. Storing and processing had to be able to scale. We had a requirement from the IT director to re-use existing monitoring, logging, an own instance of user management – and with this anyone in the world should be able to become a user. You as a researcher should be able to collaborate with anyone as long as you are allowed to. Of security concerns we had to make our own IDP, but

> we were to re-use UiO's IDP in a separate instance. Other requirements... it should be cost efficient to manage/operate and control of the dataflow in/out – a user should not necessarily be able to extract data just because they would be able to insert it (technically hard task). Later on researchers came with requirements such as: easy to use (as it was not easy to use in the beginning), self-service, data collection and more."

With a rapidly growing user base, TSD reached project number thousand in January 2020. This increase of unique projects brought innovations and complexity that would impact TSD. When TSD launched the new version of the platform, it was supposed to be secure, scalable, maintainable, and user friendly. Those criterias were met, even when challenges arose. The new architecture was created by setting up API layers using REST based API between the different components, shifting it towards a layered modular system rather than isolated monoliths.

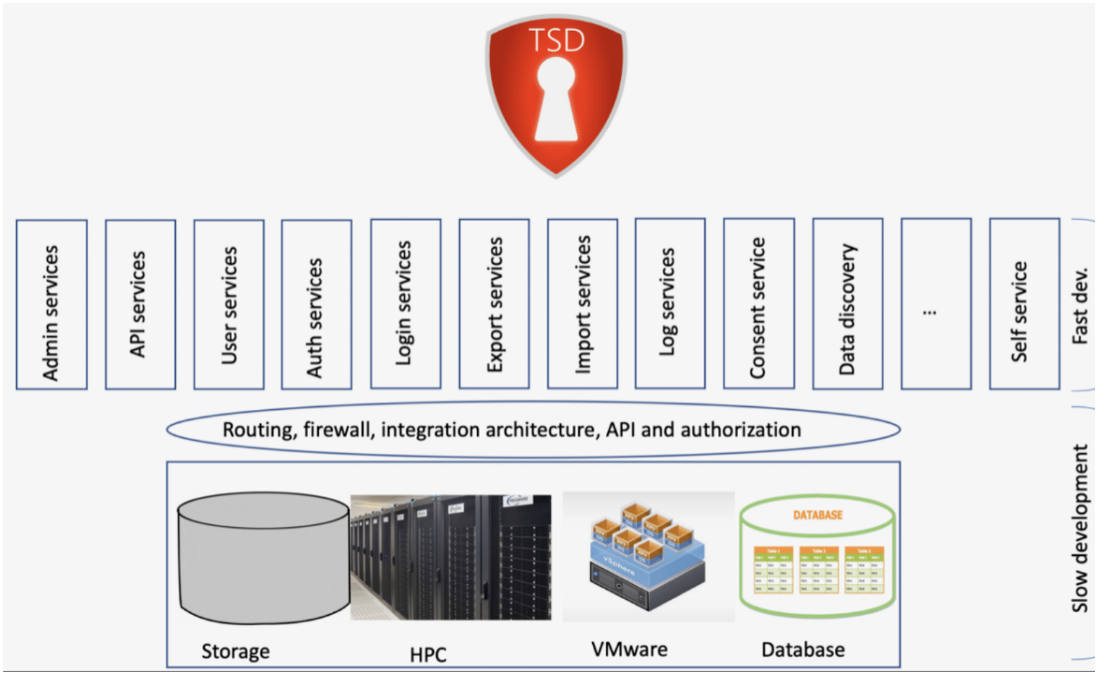


FIGURE 6.5: TSD layer overview

Figure 6.6 shows how their technologies are separated by dividing them into layers. Looking at it from the bottom and upwards, the layers can be backend, middleware, and frontend. An example of their modular and loosely coupled architecture is their SAPEIN architecture, which has to be secure in addition to these two properties.

SAPEIN - Secure API for eInfrastructures

For the access part of the system, USIT has been using Cerebrum as their Identity and Access Management system (IAM). In this research platform, the importance of people's roles in projects is very high. Different roles need to have their own set of allowed actions. When TSD offers new services and features, the IAM has to be able to scale with it. Cerebrum is used by the rest of the systems at the University of Oslo, and works well with integrating a wide range of systems. However, Cerebrum could not offer the features that were needed in TSD and contained many legacy technologies.

> "UiO has a lot of disparate systems, which they need to integrate in the Identity and Access management systems. TSD has fewer ones."

Another thing Cerebrum lacked that was needed in TSD is the notion of multi-tenancy. Adequately explained by the section leader, the case with multi-tenancy in TSD is:

> "Every research project is its own tenant all user objects and group objects exist within these tenants and in TSD you can't really bring in those semantics later on. You need to address it from the ground up. And that has been and awkward fit from the start. Trying to layer this multi-tenant semantics on top of something that is actually designed for a single tenant, sort of one organization of user accounts that belong to one organization. So, I mean the story is: We're growing fast, we need new features, we are capable ourselves."

This resulted in the development of a new IAM for TSD, which also had to be implemented at the same time Cerebrum was being phased out.

> "As long as you don't create applications that store data, and you have it in TSD, then it is solved by the architecture and you can have agility around it. "

In facing the challenges of providing flexible, secure, and extensible data transport and data management services, the TSD team at the University of Oslo designed and developed SAPEIN. SAPEIN is a reference architecture for implementing secure multi-tenant REST APIs for eInfrastructures, which can also be seen as a solution for replacing Cerebrum. SAPEIN will provide:

- API semantics for offering multi-tenant services.
- A secure and modular architecture.
- Flexible model for implementing access control.
- A development framework for the implementation of application servers and API clients.

> "The architecture is designed to centralise authentication and authorization in such a way that new application servers can reuse existing resource access control mechanisms. The implementation philosophy is that application servers should be able to focus on managing research data, while using access control services provided by the API infrastructure. In this way, new application servers add core data management functionality in a piecemeal and independent fashion without having to reinvent access control mechanisms each time. At the same time, API clients can implement higher level services by composing lower level API functionality."

SAPEIN architectural components also show us what some of the core Heavyweight IT of TSD consists of:

- External- and Internal Proxy: (External) Responsible for the incoming TCP traffic to from external networks, and routes the traffic to different servers at HTTP level. Internal Proxy acts as a gateway to the API from inside tenants.
- Data stores: Data Store A represents a central data store and Data store B represents local data stores inside tenants.
- Main servers: Authnz server is for Authentication and authorization servers, which is connected and has access to the Credentials and IAM DB. App servers, have access to both central and local Data stores.
- External- and Internal brokers: Messages from the App server gets queued in the internal broker, and can be configured to go through the external broker.
- Tenants: Can also be seen as the different projects within TSD, where researchers can access their data.

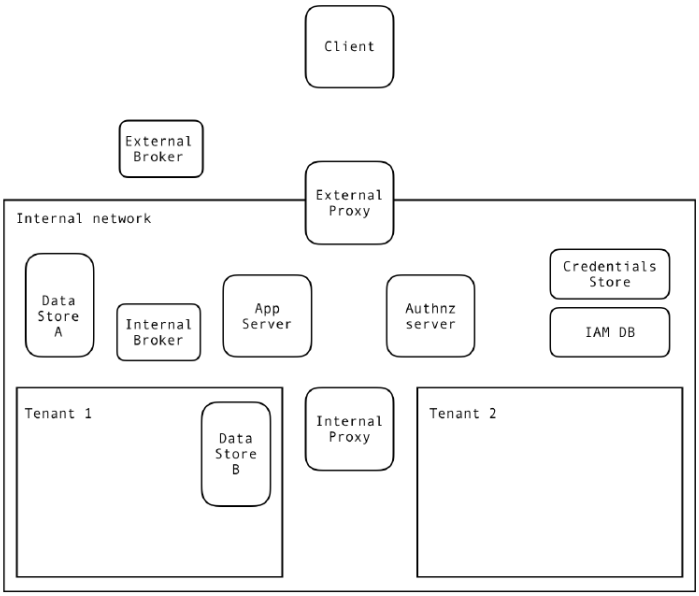


FIGURE 6.6: Overview over components in the SAPEIN solution

Clients cannot authenticate against the API without ann API key, which they will get after registering with the API. In SAPEIN, authentication requests are requests for specific access tokens in OAuth2 style, making it simultaneously authentication and authorization.

> "SAPEIN, being an API architecture, also provides a foundation for developing a rich ecosystem of clients which offer high level research services, which can b implemented by composing API functionalities."

Interoperability with other systems

A newer, more modern architecture using APIs also resulted in TSD being able to better communicate with external actors and systems. Users of TSD stem from the health sector and research environments outside of the University of Oslo.

> "Bergen, Trondheim, HUNT and NTNU there is already a consensus for communication through REST for moving data between each other."

TSD already communicates with the extensive research environments mentioned above, and there are plans to make all these systems move data between each other.

> "Now we see that we deliver mobile application to the researchers, and lately we took in a mobile app made by "Luftambulansen". When they made the application, they never considered if they had a secure back-end to store their data before it sent to the hospital. So then the hospital came to us and asked for us to host this app from luftambulansen. We said yes. Because we also run a mobile management system, so we have also distributed mobile phones, tablets in the ambulances in oslo and ambulance helicopters. "

Using a particular instance as an example, we get to see how the services provided by TSD can be used to develop applications for actors outside of the research domain.

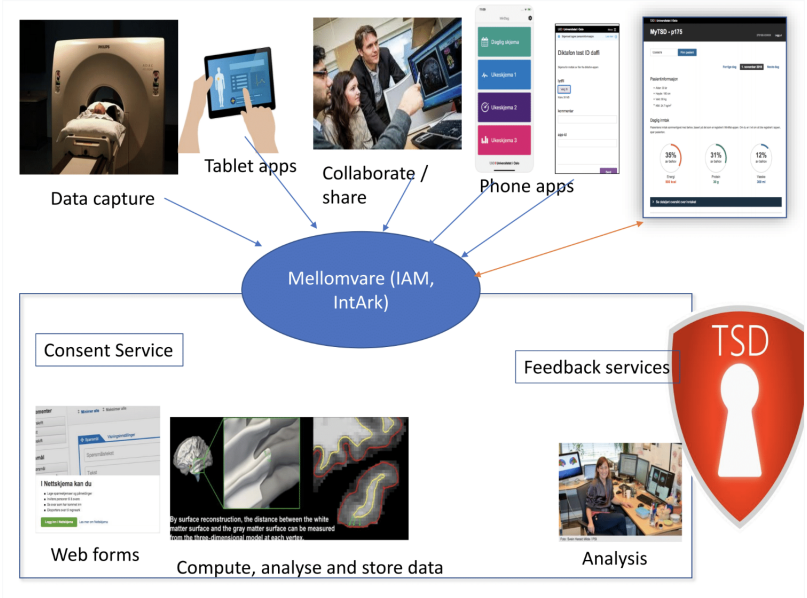


FIGURE 6.7: How applications can be built on existing services.

Figure 6.7 shows how one can develop applications using the services provided by TSD through the middleware. Among the middleware, we notice IAM being mentioned in the figure, which again can be related to the SAPEIN architecture. This helps strengthen the modularity and loose coupling of the systems architecture.

6.4 Summary

Summarizing this chapter, we presented the empirical evidence according to the framework created by using the concepts from Bloomberg (2013). Table 6.4 presents short summaries of the sections in this chapter. In the next chapter we will discuss these findings in relation to the literature, in order to answer the research question.

Topic	Summary
Processes	Processes in TSD are mapped and discussed. Among these are three processes that are frequently mentioned by interviewed users of TSD. Data analysis & processing, securely storing data and developing applications.
Services	Services in TSD are developed to support processes and change. They are all developed with security in focus, because research data have sensitive data within them. In addition to security, the platform focuses on self service and re-use of services. Nettskjema is an example of this: Users can configure their own "nettskjema" as they want, and also develop applications using nettskjema as a backend. SOA is used as the main methodology when developing services in the platform.
A loosely coupled and modular system	TSD moved from a non-elastic cloud solution to a modern, layered and modular architecture. By using an API centered architecture they obtained loose coupling between the components of the system. The layering can be visualised by presenting the system in layers, where a middleware in one layer dictates the interactions between the technologies in the other layers . This allows research groups to configure and develop their own services by using already existing functionality through boundary resources. SAPEIN is one of TSDs solutions for creating an effective API architecture that promotes the notion of loose coupling. It works as the middleware that handles communication and security within the system. This was a replacement of an IAM that was replaced, due to the lack of modern technologies and increasing complexity that brought more challenges.

TABLE 6.4: Summary of Findings

Chapter 7

Discussion

This chapter discusses the literature and empirical findings in order to build a foundation for answering the research questions. This study aims to understand what makes an architecture agile and how a platform architecture can be agile within a research domain. In light of Bloomberg (2013)'s discussion regarding agile architecture, we identified a case that suits such an investigation. This case will help us move towards an agile architecture for research platforms. The topics that we will discuss in this chapter are related to agility, architecture, loose coupling, services and processes. The research question we present is:

- How can we establish an agile architecture for a research platform?

By analyzing the empirical evidence through our theoretical framework, we saw that by strengthening the focus on processes and services, and creating a modular architecture that enables flexible development, a system architecture becomes more agile. The following two sections will discuss how these can help in establishing agile architecture in a research platform.

7.1 Strengthening the focus on processes and services.

Looking back at the research platforms mentioned in 2.2, we can see that researchers need a system to accommodate the processes of analyzing-, storing- and collecting of research data. In TSD, they uncovered that researchers also want to import data collected from their facilities and share the results.

These processes are supported by the services provided by the system owner and researchers of the system. TSD promotes the notion of self-service and service reuse through its flexible services. Reusability supports service composition within a service-oriented system (Hock-koon and Oussalah, 2010), which enables researchers to develop their own services by using existing services. Their focus on reusing services to create applications can be related to the SOA methodology (Feuerlicht and Lozina, 2007). This strengthens the flexibility of services to support the different processes (Bloomberg, 2013) of researchers in TSD. Loose coupling and weak interdependencies between services in the system are among the concepts that allow this.

Bloomberg (2013) mentions the need for processes to be flexible in agile architecture. The desire for flexibility within a business process, or research process in a research domain, raises the tension between flexibility and control. When working on research data categorized as sensitive data(EuropeanCommission, 2020b), there has to be strict control over how this kind of data is handled. Because of laws and regulations regarding sensitive data, processes focusing on sensitive research data require strict control. During the establishment of a research platform, there has to be a business process management that discusses the degree of control and flexibility that goes into each of the platform's core processes.

The researchers' activities may change over time or between research contexts, and a system that responds to change needs services that can support these changes. Bloomberg (2013) presented service-oriented architecture (SOA) as an enabler of flexible services, and the empirical evidence points out concepts from SOA, such as reuse of services and self-service, as factors for flexible services. Middleware such as an ESB has been presented as a layer to promote loosely coupled service interaction by Oracle (2020). Bloomberg (2013) argues that ESB-based architectures often focus on integration rather than the actual services when they are supposed to have a supporting role. On the other hand, we can take from this the use of a layer that promotes the loosely coupled interactions between services. To better understand the flexibility of loosely coupled services, we need to discuss how the architecture of TSD reinforces this.

7.2 Creating a modular architecture that enables flexible development.

Initially, TSD was designed and worked as a non-elastic cloud solution, much like isolated monoliths. With this solution, a growing user base and user needs would create increasing complexity similar to the tightly coupled IT Silos (Bygstad and Hanseth, 2015) and would restrict innovation and collaboration in the TSD's ecosystem. To be able to scale with the growing user base, TSD shifted towards a new architecture. A platform-cloud-like solution, as it would come to be termed, where the users would be able to develop their own services by using platform resources and reusing existing services.

Figure 6.6 presents a model demonstrating how this layered and modular platform architecture looks. The layering of the architecture is illustrated by the way infrastructure, boundary resources, and services are separated from each other. Using the technical definitions from Baldwin and Woodard (2008) and Tiwana (2014), the infrastructure (consisting of Storage, HPC, VMware, and Databases) of the system acts as a platform core, in which the core functionality exists. On top of the infrastructure lies API layers that allow researchers and other external users to use the platform functionality. This acts as an interface to the platform core (Tiwana, 2014) and the boundary resources (Ghazawneh and Henfridsson, 2013) of the platform. On the top

layer, we see the complements of the platform in the form of services and applications. These are either developed by the TSD team themselves or by the users in utilizing the resources provided in the middle layer. A platform architecture such as this promotes the notions of both modularity and innovation in the form of service and app creation by the resourcing (Ghazawneh and Henfridsson, 2013) of the functionality provided by the platform core.

Platform Layer	TSD
Platform core	Infrastructure and core functionality of the TSD system. These includes Hardware for analytics and processing, Databases, Storage and VMware.
Boundary resources (Interfaces)	IAM, middleware and integration architecture. The interfaces that enable use of functionality within the infrastructure.
Complements	Applications and services developed by either the TSD team or researchers and users.

TABLE 7.1: Comparing TSD to the platform layers from Tiwana (2014) and Baldwin and Woodard (2008)

In TSD's platform solution, there is a specific focus on sensitive data that has remained undiscussed in the existing platform literature. As TSD works with data classified as Red and Black data, they are obliged to follow the regulations regarding the sharing of sensitive data according to LSIS (from 6.2) and the EU (European-Commission, 2020b). Ghazawneh and Henfridsson (2013) discusses securing as the degree of control that the platform owner has of the platform's resources. It was found necessary to use Cerebrum, similarly utilized at UiO, as a securing mechanism for resources in the platform. The consequences of using Cerebrum as an IAM for TSD were that it was a system filled with legacy technology and integrations to many of the university's systems. As a result of implementing Cerebrum, the pace of service development within the TSD platform would slow down over time. Which would result in a non-modular architecture with inflexible development.

However, a solution to this was implementing a new REST-based API architecture that would centralize the access and authorization so that application servers can reuse existing access control mechanisms. The architecture, illustrated in figure 6.7, would replace Cerebrum and lessen the interdependence between TSD and UiOs integrated system. It would also enable the development of services at a higher pace than they have been able to before. The new architecture also supported multi-tenancy in research projects, making it easier to operate on different projects separately. This solution has managed to

meet the security requirements of distributing sensitive data in the platform while making resources accessible through API.

As the new architecture made the system more scalable and TSD acquired a more resource-efficient way to offer the platform functionality, we can say that the system could respond quickly and efficiently to changes within its environment. By this, we mean that they could design a flexible system to respond to changes, strengthening the notion of business agility (Bloomberg, 2013). Using a platform foundation for their system, they managed to meet their users' needs in terms of continuous development of services and features that would support their research processes. Furthermore, it served to be scalable and changeable. The emphasis of a layered and modular API architecture made the components of the system more loosely coupled and easier to change and transform.

We see that the change to a platform foundation of the systems architecture promotes the notion of business agility. In light of Bloomberg (2013), it would mean that the system is transformed into an agile architecture. TSD as a platform needs to develop and maintain services continuously, and to do that they need to have an architecture that supports this pace of development. Through continuous development and delivery, they continuously meet researchers' needs for services and features. By offering functionality through interfaces such as APIs, they can let the researchers develop services themselves and reuse existing services to adapt to their research processes. This also shows the flexibility of the services by adapting to the needs of research projects within the TSD platform. To provide flexibility for both services and processes in TSD, they had to develop a loosely coupled and modular architecture that would allow the use of core functionality in a secure way. By secure, we mean the use of mechanisms that handle research data according to the regulations of the EU and Norway.

Challenges affecting agility in research platforms

We explained how TSD was built as a research platform and had agile architecture traits in relation to literature from Bloomberg (2013). These were traits such as business process management that focused on user activities, services, and flexibility to adapt to changes in the processes they support and an architecture that can answer changes efficiently. For the latter part, we noticed that the system's architecture had to be changed a number of times before it became what we call a modular and loosely coupled architecture. We can look at these as challenges that need to be addressed when establishing an agile architecture.

If we look at the very first version of TSD, we can see that it was not designed to scale with the increase of users and projects. A way to answer the need for a desired future state of business processes and IT systems would be through the use of Enterprise Architecture (EA) (Tamm et al., 2011). EA works as a strategy to move from one state of the architecture to the desired state. However, Bloomberg (2013) argues that in addition to EA not being a

technical architectural representation, its goal to reach a desired state works against the principles of agility. The reason being that having a desired end state crashes with constant business change, the reason for business agility. On the other hand, articles such as Perez-Castillo et al. (2019) mention EA as a technology-driven and continuous-change process. Managing an EA to set continuous change as the main principle might help map processes and IT systems within one's enterprise to be flexible in terms of change. An enterprise in this context being the organization developing and maintaining TSD. A research platform context is referring to the organization providing services to researchers.

In the last architectural changes of TSD, the platform implemented an IAM that worked as the primary security link between users and the core functionality of the platform. The use of Cerebrum can be looked at as both a digital option and digital debt. It was a digital option because it was used at the UiO and was a working IAM that TSD could utilize. As the platform grew more significant and more complex, Cerebrum could not offer the functionality TSD needed and would hinder their scaling and growing platform. The case study from Rolland, Mathiassen, and Rai (2018) discusses how digital debt can become a hindrance in an organization's development of a digital platform, but also create new options on the other hand. As Cerebrum would be a hindrance for further innovations within the platform's ecosystem, the TSD group sought to develop its own solution, SAPEIN, that fit the organization's architecture better. This was a digital option in terms of being an opportunity to increase the organization's value proposition of work processes (Rolland, Mathiassen, and Rai, 2018). Digital debt, while being planted both willingly and unwillingly by an organization, can hinder the growth of the platform. Technically it can also introduce tight coupling and interdependencies (Rolland, Mathiassen, and Rai, 2018). In turn, this might affect the innovation and speed of development of a platform, resulting in conflicts within the agility of its architecture. Managing a platform's digital options and debt in relation to agility will have a role in how the system answers to future change.

7.3 The fundamentals of Agile Architecture in a research setting

In light of Bloomberg (2013) and the empirical evidence, we saw that there are characteristics of an agile architecture within a platform foundation and challenges in research platforms that conflict with agility.

While tightly coupled monolith systems can provide high performance and quality, the changes made to components in these systems affect the rest of the system (Yoo, Henfridsson, and Lyytinen, 2010). Thus for a system that promotes business agility, one will need a modular and layered system that focuses on loose coupling between modules. The loose coupling that a modular system brings allows for more flexibility in terms of change of components and scalability (Yoo, Henfridsson, and Lyytinen, 2010). A platform

architecture supports the layered (Tiwana, 2014) and modular (Baldwin and Woodard, 2008) concepts mostly through their stable interfaces. The empirical evidence shows that having an API focused architecture provides more flexibility and less interdependence through making the components more loosely coupled. Given this, a platform architecture could use APIs to decompose their system into modules, providing an agile structure that responds to changes. This also supports continuous development by resourcing the platform functionalities through the use of API (Ghazawneh and Henfridsson, 2013).

From the case study, we learned that a research platform, such as the one in the study, provides resources and services that allow researchers to work on research data. Working on research data consists of analyzing and processing-, collecting-, importing-, sharing-, and storing research data in a secure facility. Besides, the platform architecture allows innovation and collaboration within its ecosystem (Tiwana, 2014). Platform literature such as Tiwana (2014) focus on the concepts of multisidedness and network effects for the growth of platform ecosystems. These commercial takes are not as presented in the research domain, but the empirical evidence shows that a research platform grows through the collaboration of both the platform owner and researchers. A system where researchers can both use services for research processes or develop their own, using resources provided through interfaces promotes innovation(Ghazawneh and Henfridsson, 2013) and service orientation. Table 7.1 shows a comparison between architectural platform characteristics from platform literature and TSD. This comparison can be used as an indicator of how research platforms are built.

A research platform benefits from the notion of agile architecture, as it can help handling different research projects from various research fields. Responding to changes in the ecosystem is essential for a platform and includes adapting to all the research groups using the research platform. Meaning that a research platform should have flexibility regarding processes, services and architecture in order to better serve users from different research fields. Also, an agile architecture supports research projects that are supposed to be established quickly, and those that will be developed over time.

Given the knowledge we have obtained, an agile architecture is a structure that promotes efficient response to changes and continuous development of services. Using the literature of Bloomberg (2013) and gathering empirical evidence, we found that a layered and modular architecture with loose coupling between components enables efficient change within the system. In addition, this type of architecture can also enforce security mechanisms in a constantly changing environment that supports numerous of research projects. However, research platforms that deal with sensitive data need to meet the requirements given by laws and regulations. A cloud solution such as Platform as a Service (PaaS) could be a minimum requirement in delivering a platform solution to researchers. Looking back at the privacy surrounding sensitive data in research, one needs to discuss whether a cloud solution

is feasible. The empirical data mentioned an issue in terms of laws and regulations if data would be transferred and stored in regions such as the US. A solution such as the TSD can be characterized as more of a hybrid cloud, where having tighter control and security over applications while facilitating on-demand service development provides more flexibility than purely private and public clouds(Zhang, Cheng, and Boutaba, 2010).

7.4 Limitations

There were a few challenges along the way in this thesis. The scope and objectives of the thesis changed during the period of writing, resulting in several changes until we landed on a solid topic. After finding a topic to focus on, we experienced limitations regarding knowledge.

Gap in literature

When starting this thesis, we found out that the literature is regarding agile system architecture and research platforms. Within the term "Agile," we can see a gap in the literature, as the libraries contain mostly research on agile software development studies, but few on the topic of agile architecture. Through our contributions we tried to fill this gap by adding knowledge with our research. The literature of Bloomberg (2013) is also more focused on a commercialized and consumer-oriented industry and discusses organizations within the US. These do not meet the same legal obligations as European organizations, which can make the literature seem less favourable for a research platform in Europe. As a consequence of this, sources to lean on when discussing agile architecture are scarce.

Limitations of Data collection

Regarding the research method, we were planning to obtain information in documentation, interviews, and observations. While the two first methods were realized, the last was not achieved.Yin (2008) argues that observations will add new information, which, in this case, would be how the technology itself works and gives outside observers a more visual presentation of the subject in question. Observation is an activity one does on-site, an would make us more involved in the project and the people, gaining more practical knowledge and trust. There were two main reasons for this: It was planned for us to join observations of meetings, but this was only mentioned in two interviews and not followed through. The TSD team was already overloaded with work, so it would not sit right with us to keep asking for observations. The second reason was that during the last part of the data collection, COVID-19 grew rapidly and communication halted. At the same time, the TSD team was working hard on creating services regarding COVID-19 studies because of the global pandemic. .

Also, doing a case study on just the TSD platform, does not give us a generalized picture of agile architecture in research platforms. To increase validity of the study, one should try to do a study of more than one platform. However, TSD still is very relevant and a subject of study that we can learn a lot from.

Chapter 8

Conclusion and further work

The case study research's main goal was to investigate and understand how a research platform can fit the notion of agile architecture. We drew inspiration from the literature by Bloomberg (2013) when comparing the platform architecture of the case study to the notion of agile architecture. Chapter 4 describes the research methods we used to conduct this research. The results were presented in chapter 6 and discussed in chapter 7. In this chapter, we conclude the research and present suggestions for future work.

The research question was, "How can we establish an agile architecture for a research platform?". We drew parallels between the research platform TSD and agile architecture and research platform literature to answer this.

In chapter 3, we created a theoretical framework used to analyze and present our empirical evidence. Using this framework, we investigate how TSD fits into the role of an agile architecture with flexible services that support researchers' processes. Firstly, we discussed the flexibility of the processes of the researchers within TSD. There was a tighter control on processes regarding the security of sensitive data, such as import and sharing of data. Secondly, processes are supported by services developed by the TSD group themselves or by the researchers. Service reuse and self-service was a drive for the service orientation within the platform. By allowing researchers to develop their own services by reusing parts of existing services or combining them, change within research processes was supported.

Thirdly, we saw the architecture of TSD as an enabler of process and service flexibility. We identified TSD as a research platform in relation to digital platform literature such as Baldwin and Woodard (2008) and Tiwana (2014). The platform foundation presented itself as a modular and layered architecture, in which the components were less interdependent, and complexity was reduced by using interfaces. It also made the system more flexible and increased scalability, enabling the constant increase of research projects and continuous development of services. Their use of an IAM named Cerebrum became troublesome in terms of halting their growth due to legacy technology and deep integrations. As a solution, they had to develop SAPEIN, which is a secure and modular architecture. SAPEIN can be visualized as working in the layer that provides resources to the researchers while also securing them. It is a modular solution for the architecture, based on REST API,

which moved them away from an integration-heavy system over to a more autonomous and loosely coupled system.

TSD fits into the role of an agile architecture by having a structure that (1) responds to changes within its ecosystem and (2) allows continuous development of services that support processes of the researchers using TSD. An API based layered and modular architecture lessens the interdependence between components in the system. A loosely coupled system like this is more suited to constant changes than an integration-heavy and tightly coupled system. By looking at the literature of Bloomberg (2013), this also promotes business agility, which is a key to agile architecture.

Following the relation between TSD and agile architecture, we can begin to explain how an agile research platform architecture should be established. EA as a methodology is often used to map business processes and their relations to IT systems within an Enterprise. There have been discussions about how EA can make an agile architecture because of the concept of having a desired state. However, using EA to map the processes and systems within a research platform with change in focus rather than a desired endpoint will help make the system respond efficiently to change. It might also provide the desired flexibility in processes that Bloomberg (2013) discusses.

Technically, a digital platform foundation provides agility to the architecture with its layered and modular architecture. As security can be tightly controlled due to laws and regulations surrounding the use of sensitive data, it is important that the platform uses an acceptable IAM to secure resources. In a platform architecture, the IAM exists in the platform core or works as a layer that connects the core and boundary resources. An architectural model, such as SAPEIN can be used to illustrate how the IAM of a research platform can be both secure and modular. A modular architecture such as this enables loose coupling between components and services within the platform. Additionally, the loose coupling opens up for efficient response to changes in system components. In relation to Bloomberg (2013), this also leads to a composition-centric system with a focus on services, encouraging continuous development and reuse of services. Furthermore, TSD was presented as a cloud-like platform in the empirical evidence. Based on cloud computing concepts, a PaaS solution would work in favor of research platforms where researchers frequently develop and configure their own services.

In conclusion, when establishing an agile architecture for a research platform, one will need a cloud-like platform solution, where the core is based on SOA that emphasizes loose coupling—utilizing REST-based APIs to make components within the system more autonomous. Given this, the services will become flexible and support current processes for researchers and flexible processes that can change over time.

8.1 Future work

In the future, TSD should also explore the options of sharing data with other research platforms while following the regulations deciding how sensitive data can and are allowed to be shared. Collaboration between research platforms can amplify the innovation of research services while at the same time, increasing the tension between control and innovation. Also it would be exciting to see what kind of challenges emerges from this.

USIT has also made plans to release a lightweight version of TSD, called Forskningsplattformen, which is intended to have less focus on security and more on innovation. This could be an interesting platform to investigate and compare to TSD to discuss whether security regarding sensitive data halts a platform's agility. Also, research on digital platforms that are not in a research domain can extend our knowledge of agile architecture. Due to the lack of literature on the agile architecture of systems, research such as this over to more generalized platforms can contribute to extending agile architecture literature. Especially in commercialized platforms, there is a greater focus on meeting more comprehensive ranges of customers, which means that services and the architecture that supports them need to be more flexible than in research platforms.

TSD as a system, makes up for insightful case studies. Recently they have been able to use the potential of their agile system to develop a Covid-19 study that has supported Oslo University Hospital in their mapping of the virus spreading in Norway. Researchers could use this context to see how TSD can be used to respond quickly to future pandemics or other crises.

Bibliography

Baldwin, Carliss and C. Jason Woodard (Sept. 2008). "The Architecture of Platforms: A Unified View". In: *Platforms, Markets and Innovation*. DOI: 10.2139/ssrn.1265155.

Bloomberg, Jason (Jan. 2013). "The Agile Architecture Revolution: How Cloud Computing, Rest-Based SOA, and Mobile Computing are Changing Enterprise IT". In: pp. 267–280. DOI: 10.1002/9781118557006.index.

Braa, Jørn and Sundeep Sahay (Jan. 2012). *Integrated Health Information Architecture: Power to the Users*. ISBN: 978-93-81320-06-8.

– (Apr. 2017). "The DHIS2 Open Source Software Platform: Evolution Over Time and Space". In: ISBN: 9780262338127.

Bygstad, Bendik (2017). "Generative Innovation: A Comparison of Lightweight and Heavyweight IT". In: *Journal of Information Technology* 32.2, pp. 180–193. DOI: 10.1057/jit.2016.15.

Bygstad, Bendik and Ole Hanseth (2015). "From IT Silos to Integrated Solutions. A Study in E-Health Complexity". In: *ECIS 2015 Completed Research Papers* 23. DOI: ISBN978-3-00-050284-2.

Cisco (2019). *What is Information Security?* URL: https://www.cisco.com/c/en/us/products/security/what-is-information-security-infosec.html (visited on 03/10/2020).

Creswell, John (2007). "Qualitative Inquiry and Research Design: Choosing Among Five Approaches". In: *SAGE Publications*.

Dictionary, Cambridge University (2020a). *Agility*. URL: https://dictionary.cambridge.org/dictionary/english/agile (visited on 02/14/2020).

– (2020b). *Platform*. URL: https://dictionary.cambridge.org/dictionary/english/platform (visited on 02/14/2020).

Duggan, Dominic (Jan. 2012). "Enterprise Software Architecture and Design: Entities, Services, and Resources". In: *Enterprise Software Architecture and Design: Entities, Services, and Resources*. DOI: 10.1002/9781118180518.

European-Commission (2020a). In: *OpenAire*. URL: https://www.openaire.eu/openaire2020-project-factsheet (visited on 09/24/2020).

– (2020b). In: *EU*. URL: https://ec.europa.eu/info/law/law-topic/data-protection/reform/rules-business-and-organisations/legal-grounds-processing-data/sensitive-data/what-personal-data-considered-sensitive_en (visited on 12/02/2020).

Feng, L. et al. (2017). "The design and implementation of global energy interconnection digital research platform". In: *2017 IEEE Conference on Energy Internet and Energy System Integration (EI2)*, pp. 1–5.

Feuerlicht, George and Josip Lozina (Jan. 2007). "Understanding Service Reusability". In: *International Conference Systems Integration 2007*.

Fielding, Roy Thomas and Richard N. Taylor (2000). "Architectural Styles and the Design of Network-Based Software Architectures". PhD thesis. ISBN: 0599871180.

Galletta, Anne and William E. Cross (2013). *Mastering the Semi-Structured Interview and Beyond: From Research Design to Analysis and Publication*. NYU Press. ISBN: 9780814732939. URL: http://www.jstor.org/stable/j.ctt9qgh5x.

Ghazawneh, Ahmad and Ola Henfridsson (Mar. 2013). "Balancing platform control and external contribution in third-party development: The boundary resources model". In: *Information Systems Journal* 23. DOI: 10.1111/j.1365-2575.2012.00406.x.

Given, Lisa (2008). "The SAGE Encyclopedia of Qualitative Research Methods AU". In: *SAGE Publications*, pp. 714–718. DOI: 10.4135/9781412963909.

Harris, Shon (2018). *CISSP All-in-One Exam Guide*. Eighth Edition. Sage Publications. ISBN: 9781260142655.

Hein, Andreas, Markus Böhm, and Helmut Krcmar (June 2018). "Tight and Loose Coupling in Evolving Platform Ecosystems: The Cases of Airbnb and Uber". In: pp. 295–306. ISBN: 978-3-319-93930-8. DOI: 10.1007/978-3-319-93931-5_21.

HIMSS (2020). In: *Health Information Management System Society*. URL: https://www.himss.org/previous-himss-interoperability-definitions (visited on 12/02/2020).

Hock-koon, A. and M. Oussalah (2010). "Defining Metrics for Loose Coupling Evaluation in Service Composition". In: *2010 IEEE International Conference on Services Computing*, pp. 362–369.

Hove, S. E. and B. Anda (2005). "Experiences from conducting semi-structured interviews in empirical software engineering research". In: *11th IEEE International Software Metrics Symposium (METRICS'05)*, 10 pp.–23.

IBM, Cloud Education (2019). *IaaS (Infrastructure-as-a-Service)*. URL: https://www.ibm.com/cloud/learn/iaas (visited on 03/10/2020).

ISO (Feb. 2018). *Information technology — Security techniques — Information security management systems — Overview and vocabulary*. Standard 800-145. Geneva, CH. URL: https://www.iso.org/standard/73906.html.

Jøsang, Audun (Sept. 2017). "A Consistent Definition of Authorization". In: pp. 134–144. ISBN: 978-3-319-68062-0. DOI: 10.1007/978-3-319-68063-7_9.

Martini, Antonio, Erik Sikander, and Niel Madlani (Sept. 2017). "A Semi-Automated Framework for the Identification and Estimation of Architectural Technical Debt: A comparative case-study on the modularization of a software component". In: *Information and Software Technology*. DOI: 10.1016/j.infsof.2017.08.005.

Matsubayashi, M. and K. Kurata (2017). "Conceptual design for comprehensive research support platform: Successful research data management generating big data from little data". In: *2017 IEEE International Conference on Big Data (Big Data)*, pp. 4407–4409.

Mell, Peter and Timothy Grance (Sept. 2011). *The NIST Definition of Cloud Computing*. Tech. rep. 800-145. Gaithersburg, MD: National Institute of Standards and Technology (NIST). URL: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

Microsoft (2019). *What is cloud computing?* URL: https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/#benefits (visited on 02/05/2020).

Oracle (2020). In: URL: https://docs.oracle.com/cd/E28280_01/doc.1111/e15020/introduction.htm#OSBCA107 (visited on 09/24/2020).

Orton, James Douglas, James Douglas, and Karl Weick (Apr. 1990). "Loosely Coupled Systems: A Reconceptualization". In: *Academy of Management Review* 15, pp. 203–223. DOI: 10.2307/258154.

Parnas, David (Dec. 1972). "On the Criteria To Be Used in Decomposing Systems into Modules". In: *Communications of the ACM* 15, pp. 1053–. DOI: 10.1145/361598.361623.

Perez-Castillo, R. et al. (2019). "Enterprise Architecture". In: *IEEE Software* 36.4, pp. 12–19.

Reuver, Mark de, Carsten Sørensen, and Rahul Basole (Apr. 2017). "The digital platform: a research agenda". In: *Journal of Information Technology*. DOI: 10.1057/s41265-016-0033-3.

Rolland, Knut H., Lars Mathiassen, and Arun Rai (2018). "Managing Digital Platforms in User Organizations: The Interactions Between Digital Options and Digital Debt". In: *Information Systems Research* 29.2, pp. 419–443. DOI: 10.1287/isre.2018.0788.

Rossen, Eirik (2019). *API. I Store norske leksikon.* URL: https://snl.no/API (visited on 03/09/2020).

Schindelin, Johannes et al. (June 2012). "Fiji: An Open-Source Platform for Biological-Image Analysis". In: *Nature methods* 9, pp. 676–82. DOI: 10.1038/nmeth.2019.

Schwartz, Paul M. and Daniel J. Solove (2014). "Reconciling Personal Information in the United States and European Union". In: *California Law Review* 102.4, pp. 877–916. ISSN: 00081221. URL: http://www.jstor.org/stable/23784355.

Shin, S. et al. (2019). "Implementation of Research Data Platform: in the Perspective of Data Transfer". In: *2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN)*, pp. 735–737.

Tamm, Toomas et al. (Jan. 2011). "How Does Enterprise Architecture Add Value to Organisations?" In: *Communications of the Association for Information Systems* 28, pp. 141–168. DOI: 10.17705/1CAIS.02810.

The Open Group (2009). *TOGAF 9 - The Open Group Architecture Framework Version 9*. USA: The Open Group.

Times (2020). *Times Higher Education: World University Rankings 2020*. URL: https://www.timeshighereducation.com/world-university-rankings/2020/world-ranking#!/page/0/length/25/sort_by/rank/sort_order/asc/cols/stats (visited on 03/09/2020).

Tiwana, A. (Dec. 2014). "Platform Ecosystems: Aligning Architecture, Governance, and Strategy". In: *Platform Ecosystems: Aligning Architecture, Governance, and Strategy*, pp. 1–302.

UiO (2017). *UiO: About TSD*. URL: https://www.uio.no/english/services/it/research/sensitive-data/about/introduction.html (visited on 03/10/2020).

– (2020). *Tall og Fakta om UiO*. URL: https://www.uio.no/om/tall-og-fakta/ (visited on 03/10/2020).

USIT (2018). *What does USIT do?* URL: https://www.usit.uio.no/english/about/what.html (visited on 03/10/2020).

Weber, Anton and Schahram Dustdar (2012). *Haptic Systems Architecture Modeling*. Springer-Verlag Wien. ISBN: 978-3-7091-0755-3. DOI: 10.1007/978-3-7091-0755-3.

Weill, Peter and Jeanne Ross (2004). *IT Governance: How Top Performers Manage IT Decision Rights for Superior Results*. USA: Harvard Business School Press. ISBN: 1591392535.

Yao, Q., J. Zhang, and H. Wang (2008). "Business Process-Oriented Software Architecture for Supporting Business Process Change". In: *2008 International Symposium on Electronic Commerce and Security*, pp. 690–694.

Yin, R.K. (2013). *Case Study Research: Design and Methods*. SAGE Publications. ISBN: 9781483322247. URL: https://books.google.no/books?id=OgyqBAAAQBAJ.

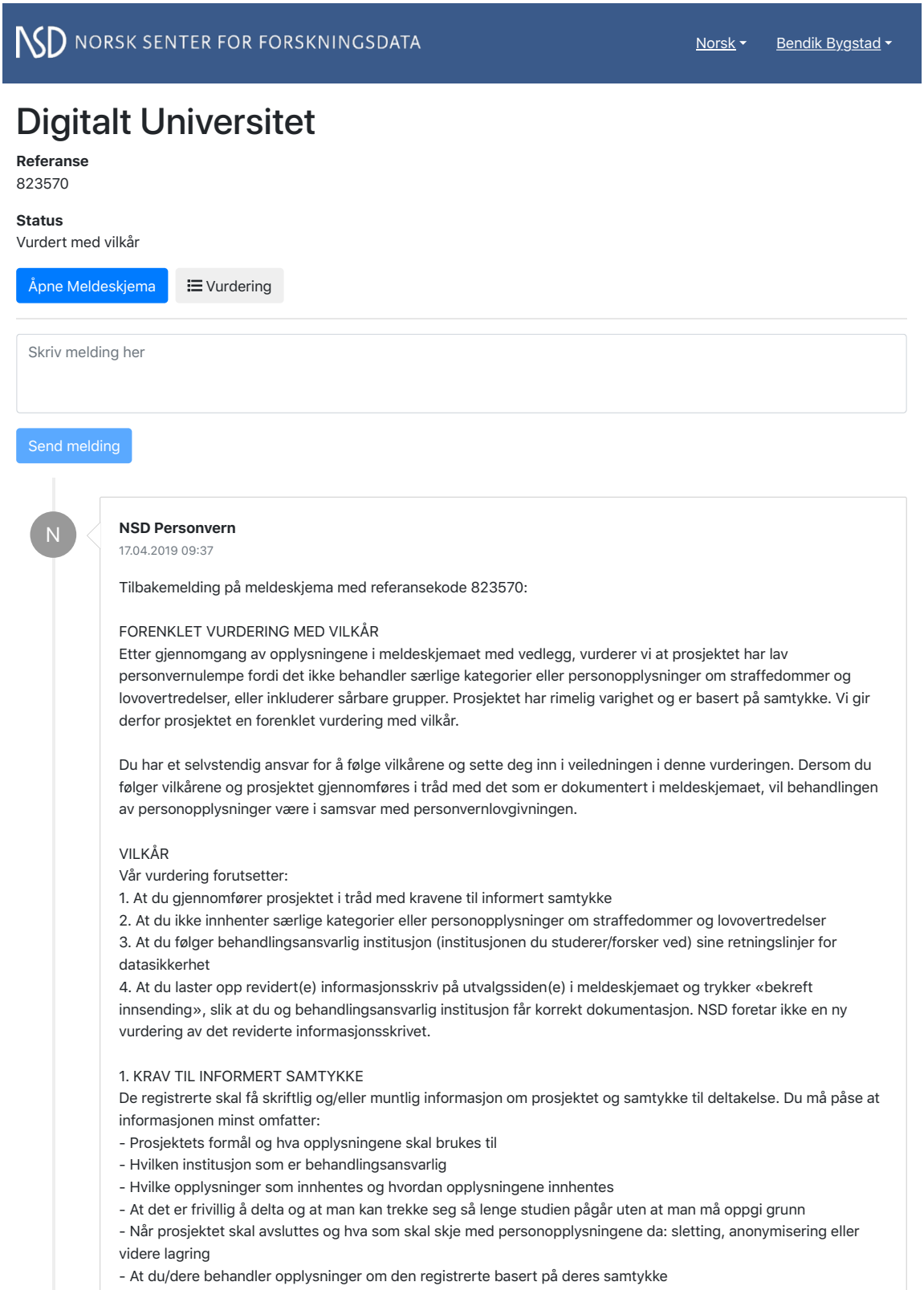
– (2014). *Case Study Research: Design and Methods*. Applied social research methods series. SAGE Publications. ISBN: 9781452242569.

Yin, Robert K. (2008). *Case Study Research: Design and Methods (Applied Social Research Methods)*. Fourth Edition. Sage Publications. ISBN: 1412960991.

Yoo, Youngjin, Ola Henfridsson, and Kalle Lyytinen (Dec. 2010). "The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research". In: *Information Systems Research* 21, pp. 724–735. DOI: 10.1287/isre.1100.0322.

Zhang, Qi, Lu Cheng, and R. Boutaba (May 2010). "Cloud Computing: State-of-the-art and Research Challenges". In: *Journal of Internet Services and Applications* 1, pp. 7–18. DOI: 10.1007/s13174-010-0007-6.

Appendix A



NSD NORSK SENTER FOR FORSKNINGSDATA Norsk Bendik Bygstad

Digitalt Universitet

Referanse
823570

Status
Vurdert med vilkår

Åpne Meldeskjema | Vurdering

Skriv melding her

Send melding

N

NSD Personvern
17.04.2019 09:37

Tilbakemelding på meldeskjema med referansekode 823570:

FORENKLET VURDERING MED VILKÅR
Etter gjennomgang av opplysningene i meldeskjemaet med vedlegg, vurderer vi at prosjektet har lav personvernulempe fordi det ikke behandler særlige kategorier eller personopplysninger om straffedommer og lovovertredelser, eller inkluderer sårbare grupper. Prosjektet har rimelig varighet og er basert på samtykke. Vi gir derfor prosjektet en forenklet vurdering med vilkår.

Du har et selvstendig ansvar for å følge vilkårene og sette deg inn i veiledningen i denne vurderingen. Dersom du følger vilkårene og prosjektet gjennomføres i tråd med det som er dokumentert i meldeskjemaet, vil behandlingen av personopplysninger være i samsvar med personvernlovgivningen.

VILKÅR
Vår vurdering forutsetter:
1. At du gjennomfører prosjektet i tråd med kravene til informert samtykke
2. At du ikke innhenter særlige kategorier eller personopplysninger om straffedommer og lovovertredelser
3. At du følger behandlingsansvarlig institusjon (institusjonen du studerer/forsker ved) sine retningslinjer for datasikkerhet
4. At du laster opp revidert(e) informasjonsskriv på utvalgssiden(e) i meldeskjemaet og trykker «bekreft innsending», slik at du og behandlingsansvarlig institusjon får korrekt dokumentasjon. NSD foretar ikke en ny vurdering av det reviderte informasjonsskrivet.

1. KRAV TIL INFORMERT SAMTYKKE
De registrerte skal få skriftlig og/eller muntlig informasjon om prosjektet og samtykke til deltakelse. Du må påse at informasjonen minst omfatter:
- Prosjektets formål og hva opplysningene skal brukes til
- Hvilken institusjon som er behandlingsansvarlig
- Hvilke opplysninger som innhentes og hvordan opplysningene innhentes
- At det er frivillig å delta og at man kan trekke seg så lenge studien pågår uten at man må oppgi grunn
- Når prosjektet skal avsluttes og hva som skal skje med personopplysningene da: sletting, anonymisering eller videre lagring
- At du/dere behandler opplysninger om den registrerte basert på deres samtykke

Appendix A

- Retten til å be om innsyn, retting, sletting, begrensning og dataportabilitet (kopi)
- Retten til å klage til Datatilsynet
- Kontaktopplysninger til prosjektleder (evt. student og veileder)
- Kontaktopplysninger til institusjonens personvernombud

På nettsidene våre finner du mer informasjon og en veiledende mal for informasjonsskriv:
nsd.uib.no/personvernombud/hjelp/informasjon_samtykke/informere_om.html

Det er ditt ansvar at informasjonen du gir i informasjonsskrivet samsvarer med dokumentasjonen i meldeskjemaet.

2. TYPE OPPLYSNINGER OG VARIGHET
Prosjektet vil behandle alminnelige kategorier av personopplysninger frem til 31.12.2022.

3. FØLG DIN INSTITUSJONS RETNINGSLINJER
NSD legger til grunn at behandlingen oppfyller kravene i personvernforordningen om riktighet (art. 5.1 d), integritet og konfidensialitet (art. 5.1. f) og sikkerhet (art. 32).

Dersom du benytter en databehandler i prosjektet, må behandlingen oppfylle kravene til bruk av databehandler, jf. art 28 og 29.

For å forsikre dere om at kravene oppfylles, må dere følge interne retningslinjer og/eller rådføre dere med behandlingsansvarlig institusjon.

NSD SIN VURDERING
NSDs vurdering av lovlig grunnlag, personvernprinsipper og de registrertes rettigheter følger under, men forutsetter at vilkårene nevnt over følges.

LOVLIG GRUNNLAG
Prosjektet vil innhente samtykke fra de registrerte til behandlingen av personopplysninger. Forutsatt at vilkår 1 og 4 følges, er det NSD sin vurdering at prosjektet legger opp til et samtykke i samsvar med kravene i art. 4 og 7, ved at det er en frivillig, spesifikk, informert og utvetydig bekreftelse som kan dokumenteres og som den registrerte kan trekke tilbake. Lovlig grunnlag for behandlingen vil dermed være den registrertes samtykke, jf. personvernforordningen art. 6 nr. 1 bokstav a.

PERSONVERNPRINSIPPER
Forutsatt at vilkår 1 til 4 følges, vurderer NSD at den planlagte behandlingen av personopplysninger vil følge prinsippene i personvernforordningen om:
- lovlighet, rettferdighet og åpenhet (art. 5.1 a), ved at de registrerte får tilfredsstillende informasjon om og samtykker til behandlingen
- formålsbegrensning (art. 5.1 b), ved at personopplysninger samles inn for spesifikke, uttrykkelig angitte og berettigede formål, og ikke behandles til nye, uforenlige formål
- dataminimering (art. 5.1 c), ved at det kun behandles opplysninger som er adekvate, relevante og nødvendige for formålet med prosjektet
- lagringsbegrensning (art. 5.1 e), ved at personopplysningene ikke lagres lengre enn nødvendig for å oppfylle formålet

DE REGISTRERTES RETTIGHETER
Så lenge de registrerte kan identifiseres i datamaterialet, vil de ha følgende rettigheter: åpenhet (art. 12), informasjon (art. 13), innsyn (art. 15), retting (art. 16), sletting (art. 17), begrensning (art. 18), underretning (art. 19) og dataportabilitet (art. 20).

Forutsatt at informasjonen oppfyller kravene i vilkår 1, vurderer NSD at informasjonen om behandlingen som de registrerte vil motta oppfyller lovens krav til form og innhold, jf. art. 12.1 og art. 13.

Vi minner om at hvis en registrert tar kontakt om sine rettigheter, har behandlingsansvarlig institusjon plikt til å svare innen en måned.

Appendix A

MELD VESENTLIGE ENDRINGER
Dersom det skjer vesentlige endringer i behandlingen av personopplysninger, kan det være nødvendig å melde dette til NSD ved å oppdatere meldeskjemaet. Før du melder inn en endring, oppfordrer vi deg til å lese om hvilke type endringer det er nødvendig å melde:

nsd.no/personvernombud/meld_prosjekt/meld_endringer.html

Du må vente på svar fra NSD før endringen gjennomføres.

OPPFØLGING AV PROSJEKTET
NSD vil følge opp ved planlagt avslutning for å avklare om behandlingen av personopplysningene er avsluttet.

Lykke til med prosjektet!

Tlf. Personverntjenester: 55 58 21 17 (tast 1)

N

NSD Personvern
17.04.2019 09:23

Kvittering på at meldeskjema med referansekode 823570 er innsendt og mottatt.

v62