Aaron Korhonen

Developing API Management Governance in Metsä Group

Metropolia University of Applied Sciences

Bachelor of Engineering

Industrial Management

Bachelor's Thesis

30 April 2024

Abstract

Author:	Aaron Korhonen
Title:	Developing API Management Governance in Metsä Group
Number of Pages:	48 pages
Date:	30 April 2024
Degree:	Bachelor of Engineering
Degree Programme:	Industrial Management
Professional Major:	ICT Management
Supervisors:	Anna Sperryn, Senior Lecturer Matti Muilu, Development and Service Manager

---

In the digital age, application programming interfaces have emerged as fundamental components driving connectivity, innovation, and efficiency. Only when managed properly can APIs be utilized to their full potential enabling systems to exchange data and functionalities seamlessly.

The thesis was conducted for Metsä Group, a forest industry company with a global presence in wood products, pulp, paperboard, and tissue paper. The goal was to improve the current state of API management governance by defining development points, core processes and roles and responsibilities.

This study is based on a theoretical foundation created as a result of the literature study, best practices relating to the topic and the findings of the current state analysis.

The outcome of the thesis is a governance model defining roles and responsibilities, development points, core processes, and the type and level of governance for Metsä Group API management was created. This model serves as a foundation for further developing and deploying API management governance.

Keywords:	Application programming interface, API management, Governance, SAP API management, Azure API management

---

Tiivistelmä

Tekijä:	Aaron Korhonen
Otsikko:	API Hallinnan Kehittäminen Metsä Groupilla
Sivumäärä:	48 sivua
Aika:	30.4.2024

Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tuotantotalouden tutkinto-ohjelma
Ammatillinen pääaine:	Kansainvälisen ICT-liiketoiminnan johtaminen
Ohjaajat:	Lehtori Anna Sperryn Kehitys- ja palvelupäällikkö Matti Muilu

---

Digitaaliaikana sovellusohjelmointirajapinnat ovat nousseet keskeisiksi tekijöiksi tietojärjestelmien yhteyksien, innovaatioiden ja tehokkuuden kehittämisessä. Oikein hallinnoituna sovellusrajapinnoilla voidaan mahdollistaa saumaton tiedonvaihto.

Opinnäytetyö tehtiin Metsä Groupille. Se on metsäteollisuusyritys, joka toimii maailmanlaajuisesti puutuotteiden, sellun, kartongin ja pehmopaperin alalla. Metsä Group on sitoutunut kestävyyteen ja edistää uusiutuvien materiaalien kehittämistä eri teollisuudenaloille. Työn tavoitteena oli parantaa ohjelmointirajapintojen hallinnan nykytilaa määrittelemällä kehityskohdat, ydinprosessit sekä roolit ja vastuut.

Opinnäytetyön tuloksena luotiin hallintomalli, jossa määriteltiin roolit ja vastuut, kehityskohdat, ydinprosessit sekä yleinen hallinnon taso. Tämä malli toimii perustana API hallinnon kehittämiselle tulevaisuudessa.

Ratkaisuehdotus perustui kirjallisuustutkimuksen tuloksena luotuun teoriapohjaan, aiheeseen liittyviin parhaisiin käytäntöihin ja nykytila-analyysin tuloksiin.

Avainsanat:	Ohjelmointirajapinta, API hallinta, Hallinto, SAP API management, Azure API management

---

Contents

List of Abbreviations

API:	Application programming interface. A set of defined rules that enable different applications to communicate with each other.
APIM:	API management. The process of developing, designing, monitoring, testing, securing, and analysing APIs.
CSA:	Current state analysis. A process management strategy that aims to identify and evaluate the current state of processes within an organization.
ITSM:	Information technology service management. The activities performed by an organization to design, build, deliver, operate and control information technology.

1 Introduction

In the digital age, application programming interfaces have emerged as fundamental components driving connectivity, innovation, and efficiency. APIs serve as intermediaries that help facilitate communication and interaction between software systems. APIs enable these systems to exchange data and functionalities seamlessly by providing standardization, protocols, and tools for accessing and managing data.

APIs provide multiple benefits for organizations looking to manage the efficiency and information security of their internal and external information systems. Some of the business benefits of using APIs include high performance, reduced human errors, enhanced customer experience and increased security. To fully harness the power of APIs in a business setting organizations need to establish API management so that it effectively serves its strategy and goals.

1.1 Business Challenge and Context

The thesis was conducted for Metsä Group, a leading forest industry company with a global presence in wood products, pulp, paperboard, and tissue paper. Metsä Group is committed to sustainable forest management and innovation, driving the development of renewable materials for various industries. As a cornerstone of Finland's bioeconomy, Metsä Group plays a pivotal role in the circular economy by utilizing wood-based resources efficiently and responsibly.

The business challenge is related to the current state of API management governance in Metsä Group. Currently the level of governance over API management and its processes is relatively low. Consequently, different business areas of the company have developed different ways of working. The lack of common guidelines for certain processes reduces operational efficiency and creates tacit knowledge within the company. With a higher level of governance, processes will become more efficient, communication clearer and information more widely shared.

1.2 Objective and Outcome

The objective of this thesis is to research APIs and API management and use the learnings to develop and improve API management and governance in Metsä Group. This is done by creating a proposal on how API management services can be developed and by defining the core processes, roles and responsibilities for API management and governance within the organization.

As an outcome of this thesis an API management development roadmap is created for Metsä Group. In addition to this the core processes, roles and responsibilities for Metsä Group API management and its governance are defined.

1.3 Scope and Outline

This thesis covers the object and outcome defined in the previous chapter, the methods of research, ways for data collection and theory about APIs and API management. The thesis starts with an introduction describing the background and motivation.

The second chapter explains the research approach, research design and data collection in detail. The third chapter consists of theory about APIs, API management and best practices related to them. The fourth chapter is a current state analysis of API management in Metsä Group containing methods and findings of the stage. The fifth chapter is a needs assessment based on the current state analysis. It contains the methods used in the stage and defines the level of governance needed. The sixth chapter contains the proposed solution to the business challenge. This chapter defines the core processes, roles and responsibilities for API management and contains the proposed governance model. The final chapter contains the summary of the thesis and a self-evaluation of the thesis.

2 Methods and Material

This chapter describes how the study is carried out by explaining the research approach, research design and the data collection.

2.1 Research Approach

The research approach explains what techniques are used in this thesis to collect data and conduct research. Qualitative research is used to get fundamental information of the topic. It is used to create the theoretical foundation for the thesis. Internal knowledge of the case company collected by interviewing API and ITSM experts working with the case company. In addition to this, internal documentation is studied.

2.2 Research Design

Research is conducted by studying materials related to API management and by conducting interviews with professionals working with the subject area for Metsä Group. As shown in figure 1 below, the objective is to create a development roadmap for Metsä Group API management and define the core processes, roles and responsibilities for API management and governance.

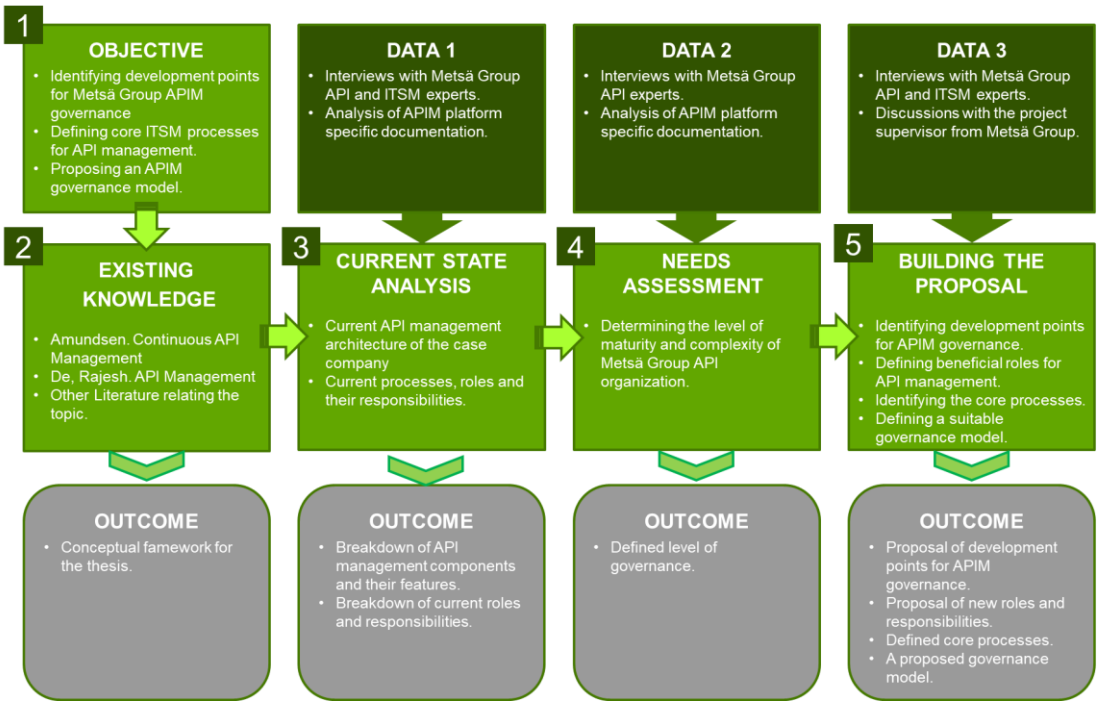

Figure 1. Research design diagram

As seen in figure 1, the first outcome is a conceptual framework for the thesis. The literature study is done as the first stage of the thesis, because it provides the basic information necessary for understanding the topic. With this the thesis can move to the next step which is a current state analysis of APIM governance in Metsä Group. As an outcome a breakdown of the API management platforms and their components, currently utilized roles and processes is created. Once the current state analysis is completed the thesis proceeds to the needs assessment phase. As an outcome the level of governance is defined.

The ultimate outcome is the development roadmap for Metsä Group APIM governance. The development roadmap contains the development points discovered during the thesis. In addition to this the most important processes, organizational roles and responsibilities for API management and APIM governance are defined.

2.3 Data Collection

Data is collected by conducting interviews with different professionals working with APIs and API management. The interviews are recorded and documented

with notes. Metsä Group also provides internal documentation such as developer guidelines that describe how APIs are managed and what technologies are used within the organization. This information is used to create the current state analysis.

After a good understanding of the current state of API management in Metsä Group is established, interviews with developers and other experts are conducted. These interviews will give a more detailed understanding of what is important for developing APIs and how API management governance can be used to efficiently support their work.

Table 1. Data collection table.

	Participant/Source	Data Stage	Data Type	Topic	Time	Documentation
1	Development and Service Manager Metsä Group	Scoping	Face-to-face interview	Project scoping	17 January 2024 120 min	Notes
2	Development and Service Manager Metsä Group	Scoping	Face-to-face interview	Project Kickoff	24 January 2024 90 min	Notes
3	Solution Architect Metsä Group	1	Teams interview	API Technologies	31 January 2024 60 min	Notes and recording
4	ICT Process Lead Metsä Group	1	Face-to-face interview	ITSM processes	13 February 2024 60 min	Notes and recording
5	Principal API & Integration Architect Vendor	1	Teams interview	API Technologies and API management	21 February 2024 60 min	Notes and recording
6	Solution Architect Metsä Group	2	Teams intervies	API Technologies and Governance	6 March 2024 60 min	Notes and recording
7	GIS and Forest Solutions Architect Metsä Group	2	Face-to-face interview	API Technologies and Governance	12 March 2024 60 min	Notes and recording
8	ICT Consultant Vendor	2	Face-to-face interview	API Technologies and Governance	12 March 2024 60 min	Notes and recording
9	Development and Service Manager Metsä Group	3	Face-to-face	Validating the proposal	27 March 2024 60 min	Notes

The table 1 above depicts which professionals are interviewed and when. It also showcases the topics and lengths of the interviews. A total of 9 interviews with 7 different people is conducted. The first two interviews are for scoping and getting the project started. The third, fourth and fifth interviews are for understanding the current state and current practices relating to API management and ITSM in Metsä Group. The sixth, seventh and eighth interviews are for getting feedback on ideas and for validating findings. The ninth interview is for validating the final proposal.

3 Literature Study

This chapter contains the topics researched in the literature study. It begins with general information about APIs and goes deeper in to detail about best practices and API management forming the theoretical foundation for the thesis.

3.1 APIs in General

Application Programming Interface is a set of defined rules that enable different applications to communicate with each other. APIs work as an intermediary layer to process data transfers between systems, enabling companies to open their application data and functionalities to business partners, external developers and internal departments. Definitions and protocols within an API help businesses connect different applications in order to make internal processes more efficient. APIs simplify design and development of new applications and services by providing developers and organizations with features such as improved communication, data monetization, system security. [IBM 2023].

In the current age the average enterprise uses over a thousand cloud applications. However many of these applications are disconnected which increases organizational overhead and reduces efficiency. APIs enable integration so that these platforms and apps can communicate with each other, automate workflows and improve collaboration. In addition to this APIs allow organizations to make connections with business partners and offer new services. APIs are often provided for free in order to build an audience around them. However, if an API provides valuable digital assets such as data, it can be monetized by charging for access to it. [IBM 2023].

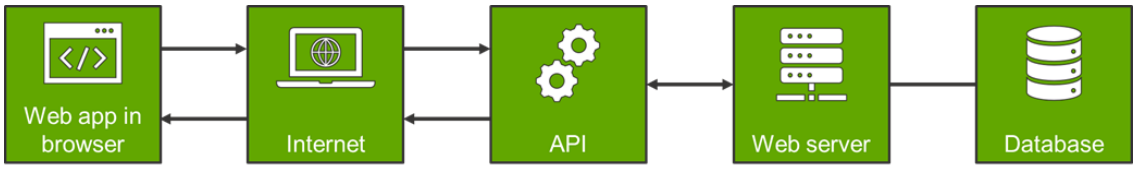

Figure 2: Basic API architecture.

As seen in figure 2, APIs provide security by separating the requesting application from the responding device and providing a layer of security between the two. This is typically done by requiring authentication credentials. APIs also provide added protection to the user by providing them a way to decline requests. [IBM 2023].

3.1.1 API Types

Most widely used APIs are web APIs that work by exposing an application's data over the internet. In general, four different types of web APIs are used which are open APIs, Partner APIs, internal APIs and composite APIs. When choosing what type of API to use, criteria should be defined. Criteria commonly taken in to account includes ease of access, scalability, level of security and the use case for the API. [IBM 2023]

Open APIs are open-source application programming interfaces that have been made publicly available to software developers. They are published and shared freely allowing universal access to consumers. The term public API is often used analogously with the term open API, but typically public APIs are more restrictive with sharing assets. Controlling how and by who an open API is used is difficult so API management has to be done properly. Open APIs provide the publisher the ability to expand their user base relatively cheaply and a chance to create revenue by licensing different programs. [Kong Inc 2024].

Partner APIs are used to connect strategic business partners. They are used when partners want to share sensitive data between the two. More often than not partner APIs are used to perform a specific task. Generally developers access these APIs in self-service mode through an API developer portal. [Kong Inc 2024].

Internal APIs or private APIs are used within an organization and hidden from external users. They are intended to improve productivity and communication across different internal products and development teams. Internal APIs use custom business logic that is generally unique to the business they belong to.

Internal APIs make it possible to share sensitive data within an organization without it getting exposed. [Kong Inc 2024].

Composite APIs combine multiple data or service APIs. They make it possible to accessing several endpoints in a single call. This can drastically reduce the number of request calls made during transactions. Building composite APIs requires taking multiple systems and their requirements in to account but can provide efficiency and ease of use when development is ready. Composite APIs are useful in cases where a single task may require information from several sources. [Kong Inc 2024].

3.1.2 API Protocols

With the increased use of web APIs, protocols have been developed to provide users with a set of defined rules. API protocols facilitate standardized information change. [IBM 2023].

SOAP (Simple Object Access Protocol) is a protocol that is used to exchange structured information between web services. It enables endpoints to send and receive data through SMTP and HTTP. SOAP APIs make it easy to share information between software components that are running in different environments. [IBM 2023].

XML-RPC is a remote procedure call protocol that relies on a specific XML format to transfer data. It allows software to running on different operating systems to communicate over the internet. XML-RPC is older than SOAP, but much simpler and uses minimum bandwidth. [IBM 2023].

JSON-RPC is a remote procedure call protocol that uses JSON to encode requests and responses. Similarly to XML-RPC it enables software running on different operating systems to make procedure calls over the internet. [IBM 2023].

REST (Representational State Transfer) is a set of web API architecture principles. REST APIs are APIs that adhere to certain REST architectural

constraints. RESTful APIs can be built with SOAP protocols, but the two standards are usually viewed as competing specification. [IBM 2023].

Traditionally APIs have referred to an interface connected to an application created with any of the low-level programming languages. Modern APIs adhere to REST principles and the JSON format and are typically built for HTTP. As a result, modern APIs are easily accessible and developer friendly. [IBM 2023].

3.2 API Governance

API governance can help to enforce checkpoints and standards through the API lifecycle. The scope of it generally includes standards, guidelines and processes that should be followed when developing, documenting, testing, deploying and operating APIs. API quality is assured with standards and principles provided by API Governance. These can include security, availability and scalability. [De and Doda 2017].

3.2.1 API governance in different phases of the API lifecycle

API Governance encompasses through all the phases of the API lifecycle. API lifecycle can be divided to five different phases which are API proposal, gathering technical requirements, building and validating the API, general availability and adoption. [De and Doda 2017].

API proposal is the first phase of the API lifecycle. This phase starts when a need for a new API is noticed. After the need is identified a proposal can be made to the API governance body of the organization in question. The best practice is to identify a real business need for the API for it to get an approval to be built. [De and Doda 2017].

After the API proposal has been approved, technical requirements for it are gathered. During this phase API architects need to work together with business analysts creating the API specifications. Technical requirements include

processes and standards for the API interface, people responsible for approval of the APIs, versioning and which back-end is used. [De and Doda 2017].

After technical requirements have been gathered and agreed on, the building of the API can be started. In this phase test scripts are created, and APIs validated for compliance to API specifications. The main goal for API governance in this stage is to define guidelines for the tools, source code repository, testing approach, review process and policies that are to be used. [De and Doda 2017].

After the API has been built, it needs to be made available for consumers and developers. This is done by publishing the API to the developer portal. Because APIs often expose valuable data, it is important to consider how and if the API will be monetized and have the terms and conditions finalized before publishing it. [De and Doda 2017].

Adoption is the final phase where API governance is needed. In this phase developers start exploring and using APIs. API governance should facilitate easy signup and onboarding of developers and new apps. The governance process should monitor how the APIs are being used. [De and Doda 2017].

3.2.2 Best Practices for API Governance

Best practices are methods or techniques that have been generally accepted as superior when compared to other alternatives. Api governance best practices aim to manage the lifecycle, development, deployment and consumption of APIs in an efficient way. These practices ensure that APIs are governed in a consistent way.

API Governance should be applied at all stages of the API lifecycle. This eliminates cases where certain things have been overlooked and end up causing issues in some phase of the API lifecycle. A core set of API governance rules should be centralized enterprise-widely. While a level of flexibility is needed to make managing and developing APIs agile and efficient, certain processes and events should follow a more strict ruleset. The goal is to reduce guess work and

tacit information by providing one source of truth that everyone within the organization can follow. [Sindall 2023].

Implementing versioning is important for keeping track of how each API has been modified through its life. Governance should enforce that all versions follow the set guidelines. [Sindall 2023].

It should be ensured, that API governance rules are met before APIs are deployed. Deploying APIs that do not follow the governance rules can cause security issues, make API management more difficult and end cause extra expenses. Following governance rules and architectural standards ensures that the APIs are complete and consistent. [Sindall 2023].

3.3 API Management in General

API management is the process of developing, designing, monitoring, testing, securing, and analyzing APIs for an organization. It is a central practice when optimizing interface usage. API management tools and platforms allow developers to scale APIs, optimize resource consumption and enforce security and rules. API management platforms can be hosted on premises, in the cloud or in a hybrid environment. [De and Doda 2017]

API management provides a range of essential advantages. One of the primary benefits of API management is the ability to provide security measures. Features like authentication, authorization and encryption ensure that APIs and the data handled by them is shielded from threats. API management platforms usually offer comprehensive documentation, sandbox environments and tools to allow developer engagement. This encourages people to collaborate with the organization providing the API. Typically API management solutions also offer monitoring and analytics capabilities. [De and Doda 2017]

When API usage grows, API management can scale to meet the demand. Scalability ensures that APIs remain responsive and reliable during periods of high traffic. API management solutions provide tools needed for governance.

Enforcing API management governance ensures compliance with industry regulations, internal standards and data integrity. [De and Doda 2017].

3.4 Best Practises for Implementing API Management

Implementing API management practices effectively is crucial for organizations seeking to harness the full potential of their APIs. Successfully implementing API management requires understanding of the organizations goals and needs. Therefore successful implementation requires setting goals, identifying success criteria, defining roles, responsibilities, processes and determining how much responsibility should the organization owning the APIs take. [Braiser and Matheny 2020].

3.4.1 Setting Goals for the API

During the first phase of implementing API management, goals for the APIs and API management are defined and an understanding of organizational needs for APIs is created. Generally there are two primary business cases for APIs, which are value generation and cost reduction. For value generation organizations commonly look to monetize APIs directly. For cost reduction organizations often try to replace existing more costly solutions. [Braiser and Matheny 2020].

3.4.2 Identifying Success Criteria

After goals are set the success criteria for the API can be identified. In practice this means understanding the organizations scope of responsibilities and capabilities. The scope of responsibilities can be divided to three different models which are platform provider, platform operator and end-to-end API Manager. [Braiser and Matheny 2020].

3.4.3 Determining the Scope of Responsibilities

One of the most critical things to take care of when implementing API management is setting the scope of responsibilities. This should be one of the very first actions taken since it will drive all future actions and decisions. Generally the scopes of responsibility can be divided to three different models which are platform provider, platform operator and end-to-end manager. [Braiser and Matheny 2020].

In the platform provider model the organization is only responsible for providing the API management platform. Teams within the organization are responsible for different APIs. The teams create APIs and are responsible for creating and updating documentation, providing support to API consumers, creating new APIs and validating them against API design guidelines. This model has the smallest scope of responsibility, but coordination between teams needs to be done in order to ensure consistency. [Braiser and Matheny 2020].

In the platform operator model organizations take responsibility for getting APIs into the platform and for the experience provided to the API consumers. In this model the organization responsible for the API platform takes care of importing APIs, creates and manages documentation and routes support and feature requests to the development team. This is a middle-of -the road model where organizations ensure that the API experience is consistent for the consumers but does not take responsibility for the APIs. [Braiser and Matheny 2020].

The end-to-end API Manager model places most stress on the team responsible for managing the APIs and is most common in smaller organizations. In this model the organization takes responsibility for ongoing support of the hosted APIs. [Braiser and Matheny 2020].

Effectively creating and supporting an API requires knowledge of the source system and the consumer’s needs. The organization will need to create strong relationships with the development team responsible for their systems. To

successfully do this, API management responsibilities need to be clearly defined and incorporated to the API management team. [Braiser, Matheny. 2020].

3.4.4 Determining Roles and Responsibilities

Clearly defining roles and responsibilities is a key part in successfully implementing API management. Roles and responsibilities can only be defined after the scope of responsibilities has been determined because only then it is clear which roles are needed. How these roles are defined depends on the size and goals of the API organization. API roles can be divided into two main categories, API consumers and API providers. [Braiser and Matheny 2020].

Task/Stakeholder	API Development Team	API Product Manager	Business leader	APIM Platform Team	OPS Team
Create new APIs	R	A	C	C	N/A
Create API documentation	R	A	I	C	N/A
Import APIs to platform	C	A	I	R	I
Update API documentation	C	A	I	R	N/A
Support consumer developers	C	A	I	R	N/A
Fix bugs	R	A	I	C	I
Add new API features	R	A	C	C	I

R = Responsible
A= Accountable
C = Consulted
I = Informed
N/A = Not Applicable

Figure 3: API Management RACI Matrix

As seen in figure 3, API consumers are the ones using the APIs managed by the API management platform such as developers or business analysts. API consumers need to have access to relevant APIs, API documentation and administrative functionalities for their own accounts. [Braiser and Matheny 2020].

API providers are the internal users of the platform. The API provider roles in use depend on the size and goals of the API organization, but likely roles are platform administrators, platform operators, API developers, API product managers and community managers. [Braiser and Matheny 2020].

Platform administrators have the most rights and responsibilities. They can see and edit everything in the platform. This role should only be used as an escalated action to prevent accidental editing of the platform's configuration.
Platform operators can see everything, but cannot edit everything. This should be the default role within the team responsible for the API platform. Access to

administrative level of permissions should be limited to cases where changes are intended to be made. [Braiser and Matheny 2020].

API developers are responsible for creating APIs. They can upload and update documentation and view and change the configuration of their APIs.
API product managers are the owners of APIs from the business perspective. Depending on the skillset of the person assigned to this role, they can administer their APIs or just have analytics access. [Braiser and Matheny 2020].

Community managers are the team members responsible for consumer focused outreach and support. This role is only needed when APIs are targeting external consumers. [Braiser and Matheny 2020].

Table 2: Responsibilities and permissions for different roles in the APIM platform.

Permissions	API consumer	API developer	API Product Manag	Community manag	Platform Admin
Access API	X	X	X	X	X
View documentation	X	X	X	X	X
Add new API		X			X
Edit relevant API documentation		X	X	X	X
Edit all API documentation				X	X
Edit configuration for relevant APIs		X			X
Edit configuration for all APIs					X
Add new users				X	X
View user contact information				X	X
Edit own contact info	X	X	X	X	X
Edit contact info for any user					X
View analytics		X	X	X	X
Edit reports			X		X

As seen in the table above, different roles have a different permissions and responsibilities on the API management platform. The best practise is to give each role only the minimum level of access that is needed. This way the risk for misuse of permissions is lowered. [Braiser and Matheny 2020].

3.4.5 Defining Strategies and Processes

The decisions made when defining strategies and processes will affect how successful the APIM platform will be. A successful API management platform includes APIs for many tasks with many target users. Despite these APIs often being developed by different teams, collectively they represent an enterprise wide API catalogue. [Braiser and Matheny 2020].

APIs should be organized in to logical groupings in order to make managing them effective and help developers find the right ones. APIs can be grouped together in multiple ways, depending on the technologies and use cases. Popular ways include grouping by target user, by business function and by security requirements. [Braiser and Matheny 2020].

3.5 API Management Platform

API Management platform is a tool used to access, distribute, control and analyze APIs. The platform is used by organizations to centralize control over their API integrations. API platforms also help ensure high performance and security standards. API management solutions feature multiple end to end services that streamline the deployment of API integrations, simplify documentation processes and sharing their configurations among development teams. [De and Doda 2017].

Generally, three different topologies are used for API management. These are cloud API management, on-premises API management and hybrid API management. In the current business environment cloud solutions are becoming the most popular option, but all three topologies are still widely used. The approach that is chosen depends on resources, goals and size of the organization choosing the platform. [Braiser and Matheny 2020].

In the cloud only model the API gateway, API administration and developer portal are delivered as a cloud service. All API requests are routed through the cloud provider for policy to be enforced. The benefit of a cloud solution is reduced administrative overhead. [Braiser and Matheny 2020].

In an on-premises-only solution all API components are deployed on premises of the API organization. All API calls are routed through on-premises gateways. This model reduces latency of calls, but incurs operational overhead for the platform and is inefficient for managing calls between cloud hosted software and APIs. [Braiser and Matheny 2020].

In the hybrid model API administration and developer portal are hosted on the cloud, but the API gateways are deployed where needed. API calls are routed through the most logical and efficient gateway. In practice this usually means that cloud-to-cloud API calls are routed through a cloud based gateway and on-premises API calls are routed through an on-premises gateway. [Braiser and Matheny 2020].

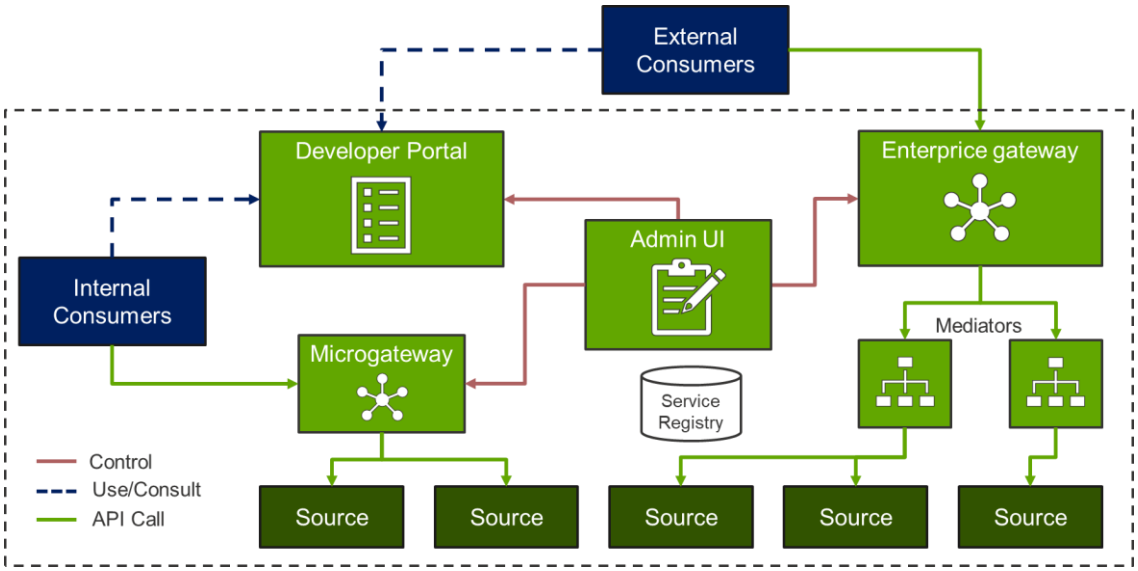

Figure 4: Basic API management platform architecture.

Basic API management platform architecture is depicted in figure 4 above. The selected approach will define the possible deployment topologies and architecture that can be implemented. When selecting an approach it is important to pay attention to the routing of API requests. If the chosen product is performing poorly or creating inefficient routes, it is recommended to extend the solution with additional components. Generally the hybrid model provides most flexibility. [Braiser and Matheny 2020].

3.5.1 Gateway

API gateway can be considered the most important aspect of an API management solution. It sits between clients and services providing a centralized handling of API communication.

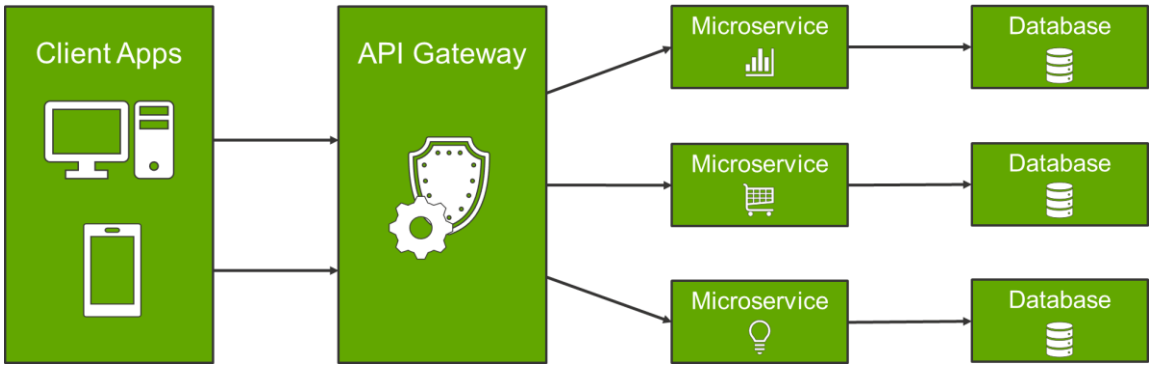

Figure 5: Basic API gateway architecture.

As depicted in the figure 5, gateways are used as an API front-end to receive API requests, enforce throttling and security policies and to pass requests to the back-end service and then pass the response back to the requestor. The API gateway provides multiple features and functionalities such as traffic management, security, high availability, meditation, ops support backend interaction and productization. [De and Doda. 2017].

The policies are enforced by the API gateway depend on the organization managing it. The type of API, it's use cases and the traffic received by it are all factors affecting what needs to be enforced. However, security and traffic management are almost always enforced by. In addition to this, the API gateway can be used for mediation, productization, to provide high availability and backend interaction. [De and Doda 2017].

Providing security is often considered to be the main purpose of an API gateway. API gateways act as a reverse proxy, sitting between backend services and the client. Using security features like authentication and authorization the gateway can accept or decline API requests from a client and direct them to the appropriate microservice. [De and Doda 2017].

Traffic management is the process of observing and controlling traffic going through an API. With traffic management suspicious traffic can be detected and blocked eliminating the potential threat it poses. In practice traffic management can be enforced by setting a spike or transaction limit, conducting SLA enforcement and by setting a concurrent users limit. [De and Doda 2017].

API mediation is used to provide a more personalized experience to consumers by providing easily consumable virtual endpoints that meet the consumer's needs. The mediation layer sits between the consumer and the API. In practice it is used to forward and translate information with features like protocol translation, message translation, message enrichment and message validation. [Bhattacharya 2023]

High availability is a feature of the API gateway that can help provide consistency. Features like clustering, health checks, load balancing and disaster recovery help APIs to provide service consistently and reliably. As an example, with load balancing requests can be routed to through multiple clusters so that the load on them is divided equally ensuring that none of them fail under heavy use. [Kong Inc 2022]

API productization allows organizations to create effective API products by making software available and consumable for customers. Making APIs consumable in a fully independent manner makes creating API products possible and relatively easy. Some features used for API productization include monetization services, billing services, third party API integration, service level contract management and CRM integration. [Clare and Powell 2021]

Backend interaction enables developers to smoothly interact with the server-side functionalities of an application. In practice it is a set of tools that permits different backend applications to communicate with each other. Some examples of these tools are data caching, middleware connections, error handling and WSDL/WADL support. [Thanh 2023]

3.5.2 Developer Portal

API developer portal is a centralized platform used to facilitate the consumption and management of APIs by developers. It works as bridge between API providers and API consumers. With API portal API providers are able to expose

and publicize their APIs and educate developers on them. It is a central hub for developers to find information and develop APIs smoothly. [Shakeer 2023].

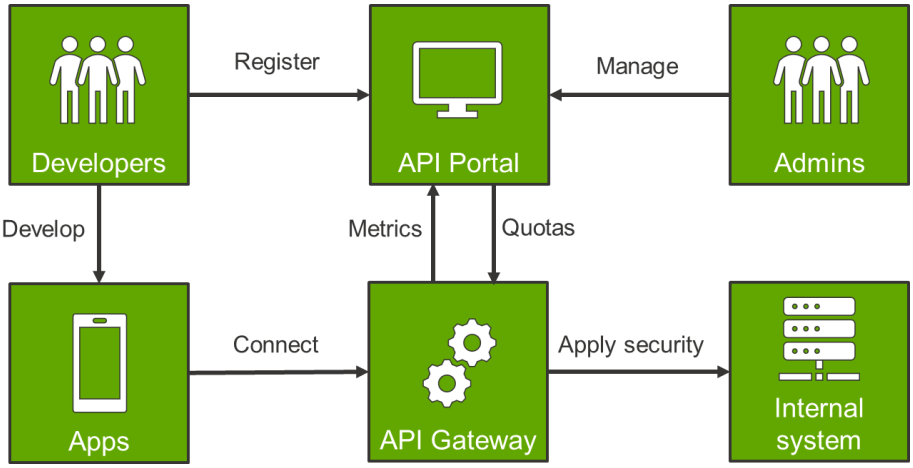

Figure 6: Basic API developer portal architecture.

As seen in figure 6, in order to get the most benefit out of the developer portal, certain roles and features are needed. Interaction with developers needs to be established, publishing APIs and apps has to be possible, API usage needs to be managed and a portal admin has to be appointed. [Shakeer 2023].

Developer interaction is facilitated by the API developer portal. It is done by providing developers with the resources needed to connect and use APIs. In practice this is done by collaborating with developers and by providing a developer console, a development environment, dashboards, and documentation. [IBM 2023].

Publishing is the process of preparing APIs or an app for the internal developer community, partners, and consumers. The product needs to be ready for use and calls from other applications when published. Versioning, permissions, management policies and contracts all need to be defined when an API or app is published. [IBM 2023].

Portal admins manages the APIs. They have access to administrative features that cannot be accessed by regular consumers or developers. They are responsible for managing the developer portal. The responsibilities of portal

admins usually include user management, gateway management, management, gateway monitoring and delegated administration. Common tools used for platform administration are support ticket systems and admin portals. [software AG 2022]

API monitoring and reporting is the practice of monitor and collecting data from an API. The collected data can be used to gain insight about the performance, availability, stress, and functional correctness of the API. In practice this can be done with custom reports, real time monitoring, ad hoc reporting and report scheduling. [smartbear.com 2024]

3.5.3 Runtime Governance

Runtime governance defines the policy actions that need to be carried out when access is requested to a particular service. The aim is to define these policies so that they promote effective collaboration, consistency, security and getting the most value out of the API. Generally, policy actions are aimed to govern security, mediation, operational practices, and compliance. [Software AG 2022]

3.5.4 Value Added Services

APIs provide opportunities to enrich customer experiences by granting access to value added services. Sometimes these services are provided for free and sometimes these are provided for a fee. Generally, these services use APIs to give access to certain data that can be beneficial. [Oyova Software LLC 2023].

3.6 API Lifecycle Management

API lifecycle management is the process overseeing APIs from their creation to retirement. The goal of API lifecycle management is to provide control over how APIs are developed, released and managed. It contains multiple individual processes that are crucial for API management, but it can also be considered a process of its own. The processes that API lifecycle management contains can vary between organizations, but in most cases it contains at least creation and

publication of APIs, version management, change notification and incident management. [De and Doda 2017].

API creation refers to the process of creating APIs. This process needs to be defined so that the correct things are taken in to account. The process contains all the phases of the creation process starting from proposing a new API and ending in the API being created. The API creation process ensures that resources are used in an effective way and that the quality of the APIs being created is consistent. The API management platform provides the tools needed to create the APIs. [De and Doda 2017].

API publication refers to the process of publishing APIs. The process contains the steps that need to be taken when publishing an API. This process ensures that APIs are built, tested and reviewed in a proper way before being published. APIs are published in the developer portal of the API management platform. [De and Doda 2017].

Version management is the process for managing updates to the APIs. This process ensures consistency and backwards compatibility between different API versions. [De and Doda 2017].

Change notification is the process of notifying API stakeholders about changes to the APIs or the platform. Changes in an API can have noticeable effects on its consumers. Therefore consumers need to be notified of any planned changes to APIs. The API management platform can be used to notify stakeholders in different ways like providing an option to join a mailing list for updates and information. [De and Doda 2017].

Incident management is the process for identifying and solving issues and errors within the APIs. This process is crucial for API management as it ensures that reaction time to incidents is quick. The API management platform should provide a way for the stakeholders to report any issues or points for improvement. In most cases this is done with a support ticket system. [De and Doda 2017].

3.7 Conceptual Framework

The conceptual framework for this thesis consists of the explanation of the most important aspects of APIs, API management and their governance. The conceptual framework for this thesis is shown in table 3 below.

Table 3: Conceptual framework.

Addressed topic from literature	References in section 3	How it was utilized
APIs and API Management • De, Brajesh; Doda, Rajesh. 2017. API Management • IBM. 2023. What is an Application Programming Interface (API)	3.1 APIs in General 3.2 Business benefits of APIs 3.4 API Management in general 3.6 API Management Platform	To provide enough knowledge on the topic of APIs and API management.
API Governance • De, Brajesh; Doda, Rajesh. 2017. API Management • Calder, Alan. 2005. IT Governance: Guidelines for Directors	3.3 API governance	To provide reference on how API governance should be done.
Best practices for implementing API managmenet • Braiser Matt, Matheny Kevin. 2019. How to successfully implement API Management	3.5 Best practices for implementing API Management	To provide reference for how API management is implemented.

Table 3 depicts the conceptual framework of the thesis. It explains what topics of literature were addressed in which section of the thesis and how it was utilized. The conceptual framework is divided into three main topics that are APIs and API management, API governance and best practices for implementing API management.

4 Current State Analysis

In this thesis the goal of the current state analysis is to define what API management components are currently used by Metsä Group and how they are utilized. In addition to this, processes related to the topic were looked at in an effort to identify which of them are clearly modelled and how they are implemented.

4.1 Overview of the CSA Stage

The current state analysis provided a great amount of valuable data for moving on to the next phase of the thesis. Many parts for API and API management governance already exists, but either are not yet clearly documented or utilized to their full potential. Overall the level of governance is adequate but clarity can be provided to multiple processes by introducing a governance model. The developer guidelines for both API management platforms contain the most crucial processes needed for operating an API platform.

The CSA stage was conducted by researching Metsä Group documentation such as developer guidelines for both API management platforms and by interviewing experts of API technologies, API Management and IT service management. Two rounds of interviews were held to experts working with both platforms. The goal of the first round was to obtain a general understanding of how the platforms are currently managed and what components are utilized. The interviews were recorded and notes were made based on the recordings. The second round of interviews was held after the findings of the first round were analysed. The second round focused on gathering opinions about the governance model and filling gaps in information remaining from the first round.

Internal API documentation was studied in addition to conducting interviews. The most important documentation for the CSA stage was SAP and Azure API management development guides. These are used by software developers as a foundation for creating and managing APIs. Among other information, the developer guidelines contained processes for API management and

development that are followed when creating and managing APIs within the organization. This information combined with the interviews made it possible to evaluate the current level of governance, which aspects can be governed in a centralized manner and which need to have platform specific governance.

4.2 Validating the CSA Findings

Three methods were used to validate the CSA findings. The first method was to compare the findings of each interview to Metsä Group documentation. This method made sure the everything was understood correctly. Documentation was also used to plan the first interviews.

The second way to validate the findings was to present the findings in the weekly steering meetings. Each week a steering meeting was held where the project counselor from Metsä Group could comment on the findings and provide additional information.

The third way to validate the CSA findings was to present them during the interviews. This way multiple experts could validate the findings and extend them by providing feedback. This method of validation was used on the second round of interviews.

4.3 Breakdown of API Management Components

Metsä Group API Management consists of two platforms, SAP API Management and Azure API Management. Due to the system architecture of Metsä Group, currently there is no practical way to only use one API management platform. Different platforms are used by different organizational bodies and business areas. Currently there is no universal guide or practice on how these platforms are governed. Instead, both platforms have their own guidelines and experts. Despite of this, the platforms had much in common in terms of processes, practices and definitions.

4.3.1 High Level Platform Architecture

On a high level the architecture of both platforms is very similar. Both platforms use an API gateway as a window to the outside of their network. Currently Metsä Group mostly uses private APIs which means that the gateway is used between their own internal systems rather than outside networks. Both platforms have a developer portal that is used to manage the platform and access data.

4.3.2 API Management Platforms

On a surface level the components and features of both platforms are very similar. However, since these platforms are provided by two different service providers, they also have differences when looked deeper into their capabilities, features and ways of managing them. Some of these differences are caused by the lack of common guidelines and governance, and some by fundamental differences between the two platform.

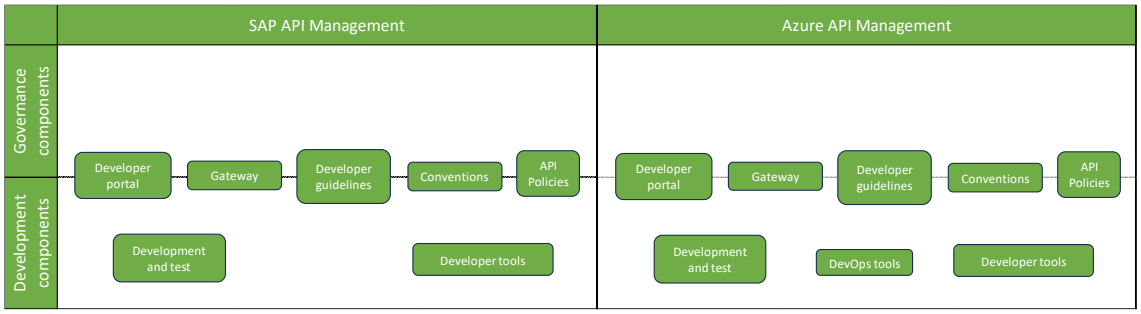

Figure 7: Components of the API management platforms

As seen in figure 7 above, most API management components are used for governance and development work at the same time. Some components are only used for development, but none of the components are exclusively used for governance.

Developer portals are used to facilitate the consumption and management of APIs for both platforms. They provide a way to consume published APIs. The users can register for the portals and gain access based on their roles within the organization. Access to the portals is managed by the administrators of each

platform. The developer portals contain development environments and documentation. Each platform has their own developer portal that is used for development work and governance. For developers they provide a platform to share information and publish APIs. For governance they provide a platform to manage all aspects of APIs such as availability and access.

API gateways are utilized by both platforms to provide security and centralized API communication. Gateways work as an entryway to the outside networks providing security and control by enforcing policies. For developers they provide a way to create secure connections and for governance they help enforce the boundaries and policies set by them.

Both platforms utilize API policies. The policies used are chosen depending on the API type and use case. How and when to use policies has been defined in the developer guidelines of each platform. API governance can utilize policies to ensure consistency, security and efficiency by providing standardization and security. This also helps with development work by providing standardized components that can be added to the APIs.

Each platform has their own developer guidelines. The purpose of these guidelines is to contain all the information a developer needs, when creating APIs. The structure and contents of the developer guidelines is similar for both platforms. The developer guidelines help API governance in ensuring API consistency. For developers the guidelines make work smoother by providing a comprehensive foundation for development work.

Each platform has their own conventions. These conventions are used to provide consistency and quality for different aspects of the API. Defined conventions vary a little between the platforms. Conventions are used by the governance to ensure standardization and consistency where possible. These features also make work for developers easier.

Each platform has their own environments for development and testing. These environments are platform specific and provide a way to create and test APIs

securely and make sure that they are tested and functional before deployment. Development and testing environments are mainly utilized by the developers.

Each platform has their own set of developer tools. For SAP API Management, all the development tools come with the platform and tools outside of that are not used. Azure API Management offers more freedom and in some cases developers can even choose what tools to use based on their preferences. Developer tools are only utilized by the developer to create APIs.

In conclusion both platforms share the same components and utilize them in a similar manner. Some differences between the platforms exist which are mostly caused by the platforms being developed individually by different teams.

4.4 Current Roles and Responsibilities

Currently both platforms have three defined roles. Despite of the roles having different names, they are very similar between the two. Both platforms have an administrator, API developers and app developers. From the developer point of view these roles are adequate and contain everything needed for building, and managing APIs. However, hierarchies and responsibilities are vaguely defined, which might cause disruptions to communication and reduce efficiency. Defining responsibilities in more detail and adding some new roles makes governance and development work easier by eliminating the previously mentioned issues.

Administrators have the highest level of access and responsibility on both platforms. They are developers or business owners who are well acquainted with the platform. And are responsible for governance and administration.

API developers are developers working on creating and publishing APIs. It is their responsibility to design, build and publish APIs to the developer portal of the platform they are using.

Application developers are developers who are subscribed to the API developer portal. Generally these type of developers can be partners or from outside of the

organization managing the APIs, but in the case of Metsä Group these are commonly internal developers.

4.5 Conclusions

The current state analysis provided a great amount of valuable data. API management in Metsä Group is well organized and all the necessary components for APIM governance already exist. Operating two different API management platforms causes some challenges from the governance point of view, but can be overcome with the right approach.

5 Needs Assessment

Needs assessment is the process of determining the needs between the current state and the desired state. It involves identifying and prioritizing the needs and problems of an organization. Needs assessment is commonly done in the first stages of a project in order to determine its scope. This chapter describes how the needs assessment was carried out. It contains the different phases of the stage, how they were conducted and the findings of the assessment.

5.1 Overview of the Needs Assessment Stage

The needs assessment stage provided valuable information regarding the scope of the proposed governance model. The main focus of the needs assessment was to define what type of governance would serve Metsä API management the best. This included defining whether the governance should be organized in a centralized or a decentralized manner and defining which level of governance would suit Metsä Group the best.

Data collection for the needs assessment was done as a part of the CSA stage. Understanding the current state of API management and it's governance within the organization, made it possible to determine where there is need for development. During the interviews, the data 2 interviewees were able to share if they had any ideas for processes, tools or other things that could make processes or their work easier or more efficient. The CSA phase revealed which processes were well implemented and which still had room for improvement. The needs assessment was done based on Metsä Group documentation, feedback from the interviewees and API literature.

5.2 Identification of Needs

The first level of needs assessment was done in the scoping phase of the project. The high level goals were defined in the beginning of the project which decided the scope for research. After the scoping was ready, API management literature and research was used to identify what is important for API management and

governance in general. Then the findings of the current state analysis were compared against the theoretical foundation in order to determine what is important for the proposal of the final stage. With this information the scope and type of the proposed governance model could be decided.

5.3 Defining the Level of Governance

Defining the level of governance was an important part of the needs assessment. It helped to create an understanding of how much control should be placed on the API governance and how much control can be left on the development team. Level of governance was defined by assessing the maturity and complexity of Metsä Group API management organization as a whole. Organizations with high maturity and low complexity can be more autonomous and have a low level of governance where organizations with high complexity and lower level of maturity need to have a higher level of governance.

Maturity was evaluated based on organizational expertise and level of documentation. The maturity of Metsä Group API management organization was evaluated as high. Documentation for both platforms is comprehensive and covers all of the crucial aspects of API management and the API lifecycle. Despite not having a common governance model, both platforms have evolved processes covering the API lifecycle from start to end. For the most part the processes are very similar and follow best practices. Interviews with experts revealed that the development teams have a high level of expertise and can be trusted to work autonomously.

Complexity was evaluated based on the amount of variables within the APIs, API management platforms and their scale. The complexity of API management was evaluated as high. Having multiple business areas and operating two API management platforms makes the whole of API management in Metsä Group complicated.

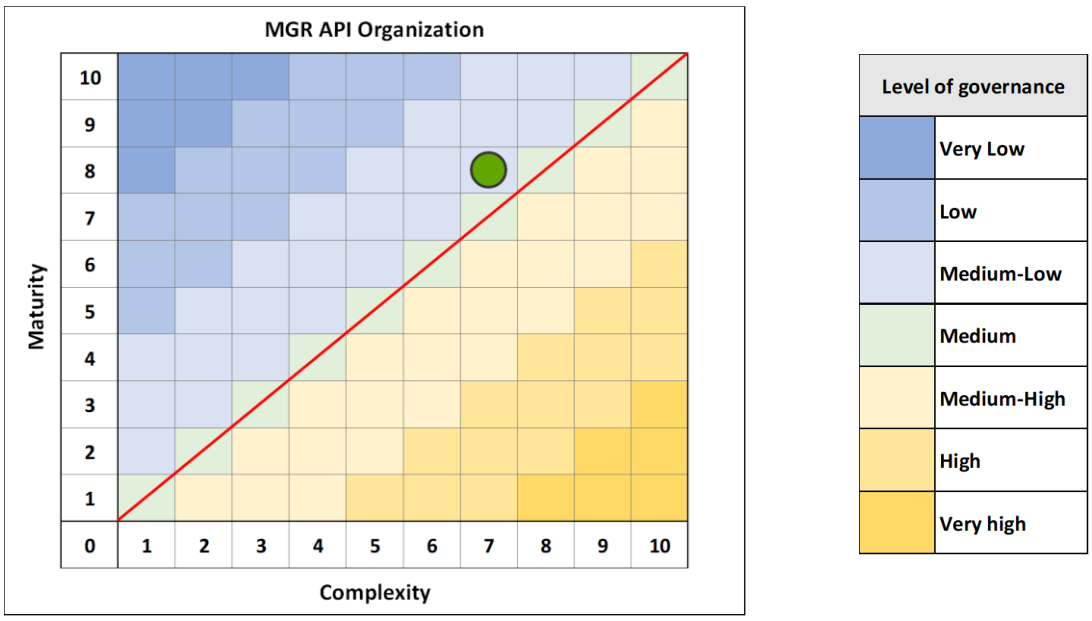

Figure 8: Chart visualising the needed level of governance.

As seen in figure 8 above, Metsä Group has both high maturity and high complexity. Maturity is slightly higher than the complexity so the level of governance was defined to be medium-low. With this level of governance autonomy is allowed for the API management organization but governance is still needed. The level of governance also depends on the criticality and importance of the area being evaluated.

6 Building the Proposal

This chapter describes how the solution phase was conducted. It begins with an overview of the stage and moves on to a more detailed breakdown of each part of the proposal.

The current state analysis provided a great amount of valuable data for moving on to the next phase of the thesis. Many parts for API and API management governance already exist, but either are not yet clearly documented or utilized to their full potential. Overall the level of governance is adequate but clarity can be provided to multiple processes by introducing a governance model. The developer guidelines for both API management platforms contain the most crucial processes needed for operating an API platform.

6.1 Defining the Core Processes for Metsä Group API management

Core processes are the processes needed for composing the main activities of an organization. For API management these are the processes that are used to create, develop, maintain and manage APIs. They extend throughout the API lifecycle providing framework on how to operate in different situations.

The findings of the current state analysis, needs assessment and theoretical foundation were used to define the core processes for Metsä Group API management. The current state analysis was used to understand what is important for API Management from the point of view of Metsä Group. The needs assessment was used to define the scope and level of governance needed. The theoretical foundation was used to determine which processes are relevant for API management on a general level. In this chapter the core processes for Metsä Group API management are defined and showcased on a general level.

On a high level the core processes for running API management are the same for both platforms. In more detail the platform specific process descriptions have some differences, but also have much in common. The core processes were defined based on the platform specific development guidelines, interviews and

theoretical foundation. As a result access management, API creation process, API publishing process, incident management, change management, version management and API monitoring were determined to be the core processes. These processes extend through all the phases of the API lifecycle and are needed to create and manage APIs in a consistent and efficient way.

Access management is the process of controlling and regulating access to a platform. In practice access management includes managing user accounts and permissions to ensure that users have appropriate access to data and features. It provides security and control for the API organization. Process for access management has been modelled for both APIM platforms.

The process for creating APIs needs to be defined so that APIs are created with consistency and in a way that fits the standards of the organization owning it. The creation process contains all the phases of API creation starting from the proposal of a new API and ending in its deployment. A properly defined creation process ensures that resources are used effectively and that the quality stays consistent. The API creation process is modelled for both APIM platforms.

The process for publishing APIs is used to make APIs available to developers. The process ensures that APIs are reviewed in a proper way before being published. As a result the quality, security and documentation of published APIs stays consistent. The process is modelled for both APIM platforms and is similar between them.

Incident management is the process for identifying and solving issues and errors. The goal of incident management is to restore normal operation of services as quickly as possible. In practice this is done by having a structured and well defined process so that response to incidents can happen quickly and effectively. This process is crucial for API management as it ensures that connections work smoothly and the amount of connection downtime caused by incidents stays low.

Version management is the process for managing updates and changes to a document or product. In practice it is done by tracking updates and numbering

different versions consistently. For APIs a version reflects a change in the signature of an API and requires developers to modify the code accordingly. The version management process ensures that changes are documented and provides consistency and backwards compatibility between different versions. The version management process has been documented for both APIM platforms and is almost identical between the two.

Change management is the process of planning, implementing and controlling changes. The goal for change management is to reduce disruptions and risks when implementing changes. Change management fosters a culture of continuous improvement by capturing lessons and best practices learned from previous experiences. Currently neither platform has an API specific change management process, but a Metsä Group common process for it does exist.

API monitoring is the practice of monitoring API traffic and health. Among other things, API monitoring can be used to see what kind of load each API is under and to give early warning signs if a connection is about to fail. It can also be utilized to track how many API calls are made by a specific API consumer when monetizing APIs. The scope of API monitoring has been defined for Azure API management, but is not currently utilized to its full potential with SAP API management.

Cost management is the process of planning and controlling the costs of running a service. Cost management is done by collecting, analysing and reporting cost data to effectively utilize resources. Currently cost management is not done for either APIM platform and their costs are seen as part of the total cost of the related integration.

Table 4: Core API management processes and their importance.

Process	Importance	Reasoning
Access management	High	This process ensures that only proper users have access to the platform with the correct level of access.
Creation process	High	This process provides the framework for creating APIs and ensures that their quality is consistent.
Publishing process	High	This process ensures that APIs are reviewed correctly before publishing and all necessary data is provided.
Incindent management	High	This process ensures that incidents are dealt with quickly and service downtime is minimized.
Version Management	Moderate	This process ensures that updates and changes to the APIs are documented.
Change Management	High	This process ensures that changes are implemented in a secure manner without disruptions.
API Monitoring	Moderate	This process ensures that data provided by the APIs is utilized efficiently to its potential.
Cost management	Moderate	This process ensures that resources are utilized efficiently.

The core processes, their importance and reasoning for the defined level of importance is displayed in table 4. The above-mentioned processes can be defined as the core processes for Metsä Group API management. All of these processes are needed for smooth and secure operation of APIs. Even though all of these processes are needed, some can be considered to be of a higher importance. API management can operate temporarily without some processes being enabled while the lack of others will cause issues quickly.

6.2 Proposed Roles and Responsibilities

As part of the proposal stage, API management roles and responsibilities were redefined. The area of responsibility for each role was re-evaluated and new roles were introduced. Since the previously existing roles were development-centric the focus was on adding more governance perspective. The goal was to define areas of responsibility more accurately. The roles were developed based on the theoretical foundation, Metsä Group APIM platform specific development guidelines and the interviews.

Task/Stakeholder	API Developer	API Product Manager	Business Owner	Application developer	Platform Admin
Approval of new APIs	C	R	A	N/A	I
Create new APIs	R	A	C	I	I
Create API documentation	R	A	I	I	I
Import APIs to platform	R	A	I	I	I
Update API documentation	R	A	I	I	I
Support consumer developers	R	A	I	I	I
Fix bugs	R	A	I	I	I
Add new API features	R	A	C	I	I
Create new apps	N/A	N/A	N/A	R	I
Managing access to platform	N/A	N/A	N/A	N/A	A/R

R = Responsible
A= Accountable
C = Consulted
I = Informed
N/A = Not Applicable

Figure 9: RACI Matrix of the proposed roles and responsibilities.

As seen in figure 9 above, a RACI matrix was used to map out the different areas of responsibility for each role. New roles of API product manager, platform admin and business owner were introduced while API developer and application developer stayed the same. The responsibilities previously belonging to the administrator were divided for the API product manager and platform administrator. The main responsibility of the API product manager is to ensure that each API product has an owner. The main responsibility of the platform

admin is to manage access to the APIM platform. The role of business owner is currently not utilized, but it might be needed in the future.

In addition to defining roles and responsibilities, access rights to different features of the APIM platform were also defined. Access is given based on the roles, so that each role only has access to features that are relevant for them. Table 5 below is used to depict which features each role has access to.

Table 5: Access to different features of the APIM platform per role.

Permissions	API developer	API Product Manager	Business owner	Platform Admin	Application develope
Access API	X	X	X	X	X
View documentation	X	X	X	X	X
Add new API	X	X		X	
Edit relevant API documentation	X	X		X	
Edit all API documentation				X	
Edit configuration for relevant APIs	X	X		X	
Edit configuration for all APIs				X	
Add new users				X	
View user contact information			X	X	
Edit own contact info	X	X	X	X	X
Edit contact info for any user				X	
View analytics	X	X	X	X	
Edit reports		X		X	

As seen in the table above, different roles have a different permissions and responsibilities on the API management platform. The proposed permissions were chosen based on the literature study and interviews with experts.

6.2.1 Breakdown of the Proposed Roles

The role of API developer was kept the same as before. They are developers who create and maintain APIs. API developers are mostly responsible for development work and repairing issues when they emerge. They can also be assigned the role of API product manager and platform admin if convenient. API developers have a medium level of access to features on APIM platform. API developer is purely a development role that does not have governance responsibilities.

API product manager is the owner of API products. Each API product should be assigned an API product manager that is accountable for its maintenance and development. There can be multiple API product managers on a platform but each API product should only have one manager. This role has a medium level

of access to features on the APIM platform. API product manager has both development and governance responsibilities.

The main responsibility of the business owner is to justify the operation of the APIs from the business perspective. The business owner has to understand the needs for the APIs and their scope. There can be multiple business owners on an API platform, but each business area should only have one owner. This is purely a governance role and no development work is expected. This role has a low level of access to features on the APIM platform. When the business owner role is not utilized, the API product managers can take over the responsibilities of the role.

The platform admin is responsible for administrative work on the API management platform. Each platform should have its own admin. The platform admin has the highest level of access and rights on the APIM platform. The person assigned to this role needs to be well experienced with the platform and stay informed about future updates and changes. This is mainly a governance role, but some development work might be included. The platform admin can also be assigned other roles if convenient.

The role of application developer was kept the same as before. Application developers create applications utilizing the APIs. Application developers can be internal or external of the API organization. Application developer is purely a development role and does not have governance responsibilities. This role has a low level of access to features on the APIM platform.

The platform admin is responsible for administrative work on the API management platform. Each platform should have its own admin. The platform admin has the highest level of access and rights on the APIM platform. The person assigned to this role needs to be well experienced with the platform and stay informed about future updates and changes. This is mainly a governance role, but some development work might be included. The platform admin can also be assigned other roles if convenient.

6.2.2 Relation Chart of the Proposed API Management Roles

All of the roles need to have some level of access to the API management platform. Some roles are accountable to others and some roles have no accountabilities.

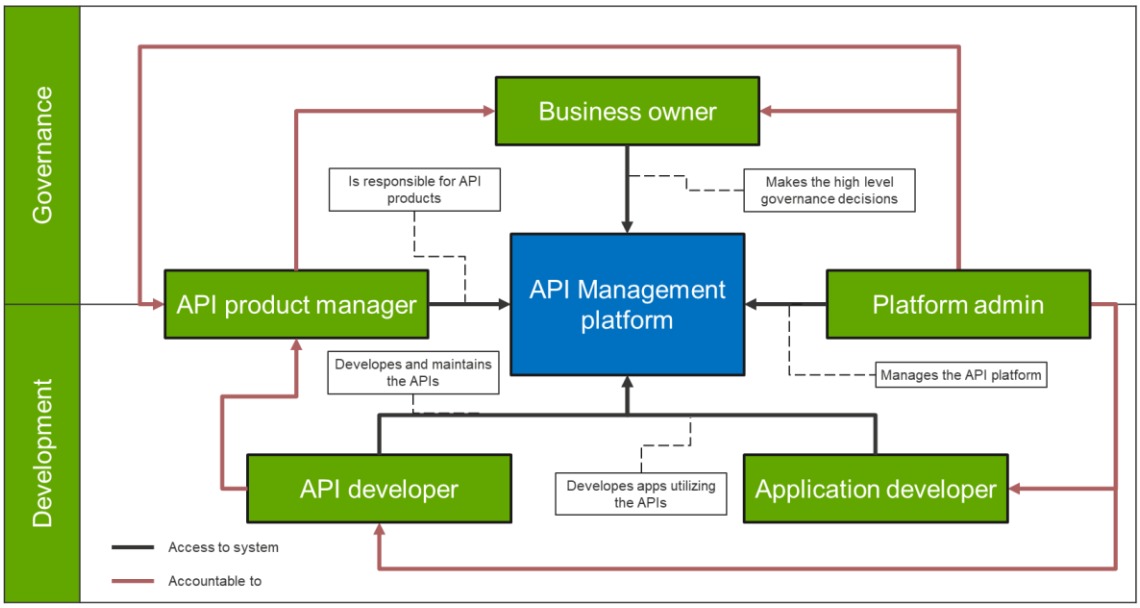

Figure 10: Relation chart of API management roles.

As seen in figure 10, API developers are accountable to the API product managers for creating and maintaining APIs. The API product managers are accountable to the business owners for successful development and maintenance of the APIs. The platform admins are accountable to all stakeholders for providing necessary access and keeping them informed about relevant information relating the platform.

6.3 Proposed Governance Model

Metsä Group has a hybrid model of API management with two platforms. Because of this a hybrid governance model was proposed. A hybrid model makes it possible for some parts to be governed in a centralized manner and others in a decentralized. This enables the utilization of existing expertise and documentation in an efficient way without making the governance model too rigid.

6.3.1 The Scope of Governance

The scope of the APIM governance model was determined based on the theoretical foundation and interviews. In order to stay within the scope and timeframe of the thesis, the proposed governance model only covers the most central aspects of API management. It serves as a foundation that can be extended on in the future. The goal was to centralize as much of the governance as possible, but also have the option to leave aspects decentralized when convenient. The scope of the governance with reasonings why each aspect was chosen are displayed in table 6 below.

Table 6: The scope of governance and reasonings.

Process / Feature	Reason for including it
Access Management	This process ensures that only proper users have access to the platform with the correct level of access.
API Creation process	This process provides the framework for creating APIs and ensures that their quality is consistent.
API Publication Process	This process ensures that APIs are reviewed correctly before publishing and all necessary data is provided.
Development Tools	Development
Version Management	This process ensures that updates and changes to the APIs are documented.
Change Management	This process ensures that changes are implemented in a secure manner without disruptions.
API Monitoring	This process ensures that data provided by the APIs is utilized efficiently to its potential.
Cost Management	This process ensures that resources are utilized efficiently.
API Catalog	API catalog ensures that all relevant information is documented and can be found with ease.
Design Standards	Design standards ensure that APIs are designed and created in a correct way.
API Policies	API policies ensure control over APIs.

The aspects of API management within the scope of governance and the reasoning for why they were chosen is depicted in the table above. These aspects cover the most important parts of API lifecycle and are crucial for successful API management.

6.3.2 Choosing the Type of Governance Model

The type of governance model proposed was chosen based on the current state analysis and needs assessment. The maturity and complexity of API management in Metsä Group was evaluated in order to find the best approach for them. Three different governance models were considered for the proposal: centralized, decentralized and hybrid.

In a centralized model governance responsibilities are placed on single governing body. This model offers high level of consistency and standardization, but lacks

flexibility. Since the two APIM platforms operated by Metsä Group have been developed individually and have their own teams of experts, this model was not chosen. A centralized governance model would not allow much platform specific decision making and would underutilize the expertise of the organization.

In a decentralized governance model governance responsibilities are divided among different teams and organizational bodies. This model offers high flexibility but can cause issues in consistency and coordination. This model requires a high level of expertise and becomes hard to manage if complexity is increased. Currently API management in Metsä Group is orchestrated in a decentralized manner. This model was not chosen since the whole of API management in Metsä Group is relatively complex, and in the long run this approach might not be the most efficient or scalable.

In a hybrid governance model elements of centralized and decentralized model are combined. In this model certain aspects are governed in a centralized manner and some in a decentralized. The hybrid model can be tailored to facilitate organizational needs precisely but might make governance work more complicated. This model was chosen as the proposal because it suited the goal of developing governance and provided the flexibility needed for governing two platforms simultaneously.

A hybrid governance model was proposed because Metsä Group operates a two platform API management model. The hybrid model makes it possible to increase the level of governance gradually while still giving experts of each platform a way to individually affect ways of governance and development. The goal for API management governance is to provide consistency and ease of development and management. In order to succeed in this goal, the governance model needs to be flexible and provide a level of autonomy.

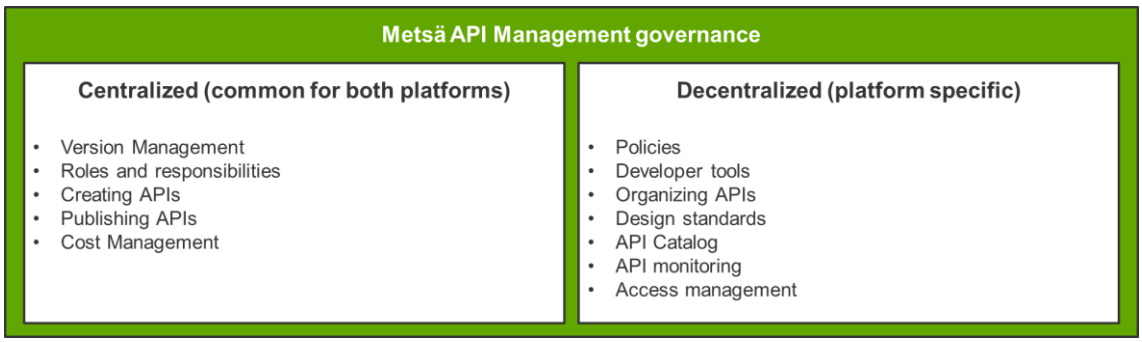

Figure 11: Centralized and Decentralized processes.

The scope of centralized and decentralized aspects is depicted in the figure above. Aspects that were already very similar between the platforms were proposed to be governed in a centralized manner. The scope of the centralized governance was kept relatively small. The reasoning for this was that that up until now the platforms have been developed individually so the transition in to a model with a higher governance would go over more smoothly when done gradually. The scope of centralized governance includes version management, roles and responsibilities, creation process, publication process and cost management.

The scope of decentralized governance included policies, developer tools, organizing APIs, design standards, API catalog, API monitoring and access management. These aspects were chosen because they are very platform dependent.

6.4 Development Points

The proposed development points were chosen based on the findings of the current state analysis, needs assessment and theoretical foundation. This chapter describes the methods used to identify and visualise the development points relating to API management in Metsä Group. Due to its business sensitivity, the actual development points are not shared in this thesis.

The development points were identified by comparing internal documentation to the findings of the theoretical foundation. This approach provided a good understanding of how the current state of API management compares against best practices. In addition studying documentation, Metsä Group experts were

given a chance to share their opinions during the interviews. The experts were able to provide some valuable insights that could have not been identified by only researching documentation. The development points were validated by displaying them to different experts during the interviews and seeing if they had any opinions about them.

Since no timeline was configured for the development goals, a hierarchical approach was chosen as a way to visualize the findings instead of a development roadmap. This way the importance of each development point was understood and could be easily placed on the roadmap once a timeframe is established.

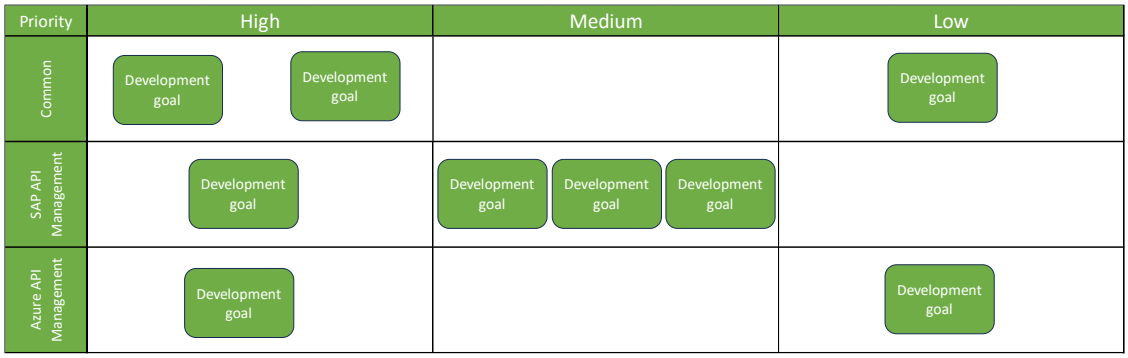

Figure 12: Reference image of the development matrix.

A reference image of the development matrix is depicted in figure 12 above. The matrix consists of three categories and three levels of priority. The categories are on the vertical axis of the matrix and show which API management platform the goal is aimed to. This approach was chosen since the platforms had some different and some common development needs. Priority of the development goals is on the horizontal axis of the matrix. Three levels of priority are used since it provides enough accuracy without making the matrix too complicated.

7 Conclusions

This chapter contains a summary, a self-evaluation of the thesis and the closing words. The chapter starts with a summary highlighting the most important aspects of each step, moves on to self-evaluation and ends in closing words highlighting what it was like to work on this thesis and what was achieved.

7.1 Summary

The objective of the thesis was to define the core processes for API management and create a development roadmap for Metsä Group API management service. The core processes were defined, a current state analysis, needs assessment and a proposal was made. The study was started by determining the scope and objective of the thesis. Based on this the research design could be created. After this the conceptual framework could be created and the thesis could proceed to the creation of the theoretical foundation.

The theoretical foundation was created by researching APIs, API management and their governance. The goal was to create a good understanding of the topic. The goal was reached and as a result the theoretical foundation could be used to guide the rest of the thesis. The theoretical foundation addressed three main topics that are APIs and API management, API Governance and Best practices for Implementing API management.

The main focus of the current state analysis was to understand the current level of API management governance in Metsä Group. In practice this was done by researching internal documentation and interviewing API and ITSM experts. With this a breakdown of the current roles, responsibilities, APIM platforms and their components was created. The main finding of this stage was that both APIM platforms operated by Metsä Group have much in common. The platforms consists of the same main components that are used mostly for the same purpose. Currently utilized roles were also similar for both platforms. In conclusion the current state analysis revealed that governance of both platforms can be easily centralized up until a certain level. The current state analysis was

used as a foundation for the needs assessment by defining what exists and what still needs to be developed.

The main focus of the needs assessment was to determine the level of governance that would suit Metsä Group well. It was determined that the a medium-low level of governance would suit well. This approach was chosen based on evaluations of the level of maturity and complexity of Metsä Group API organization. For the most part the needs of Metsä Group relating to this thesis were already known at the start of the thesis, hence greater focus was placed on the current state analysis and building the proposal. The proposal was created based on the current state analysis and the needs assessment.

After the current state was well understood and the needs assessment was ready, the building of the proposal could be started. The main focus of the proposal was to define core processes, propose beneficial roles and responsibilities and a governance model for Metsä Group API management. In addition to this some development points were identified. Eight processes were defined as the core processes for API management. These processes extend through all the phases of the API lifecycle enabling API management. Two new roles were proposed and the responsibilities of existing roles was shifted around in an effort to make hierarchies and accountabilities more clear. A hybrid governance model was proposed since it seemed to fit the current state of Metsä Group the best by providing a certain level of flexibility while still adding structure and clarity to APIM governance.

7.2 Self-evaluation of Thesis

The thesis followed the research design and achieved the set goals. The level of depth in the current state analysis is adequate and the proposal covers all the aspects that were defined during the scoping. Despite a development roadmap not being created, the development points were identified and the importance of each point was evaluated. This approach also provided the wanted results and can be easily converted into a development roadmap when the timeframe is

established. As a whole the depths and contents of the thesis are sufficient for providing a foundation for deploying the governance model.

The goal of creating a comprehensive theoretical foundation was successfully achieved. Research was conducted by exploring concepts related to APIs, API management and their governance by studying relevant literature. The theoretical framework provided a solid basis for understanding the complexities of the topic.

7.3 Closing Words

Working on the thesis was a challenging and rewarding experience. It provided an opportunity to utilize and enhance existing skills gained from previous studies. In addition to this new skills were acquired. Conducting interviews with ICT experts provided an interesting look into different areas of Metsä Group ICT. Working on the thesis provided valuable understanding of APIs, API management and ICT management in general. For Metsä Group this thesis provided a good understanding of the current state of API management and its governance. In addition to this the proposed governance model provides a good foundation for developing APIM governance to a higher level. Working on the thesis was a valuable learning experience.

References

Amundsen. Mike; Medjaoui. Mehdi; Mitra. Ronnie; Wilde. Erik. Continuous API Management, 2nd Edition. Available at: <https://learning.oreilly.com/library/view/continuous-api-management/9781098103514/?sso_link=yes&sso_link_from=metropolia-university>. Accessed 6 Feb. 2024.

Bhattacharya, Budhaditya. 2023. What is API mediation?. Available at: <https://tyk.io/blog/what-is-api-mediation/> Accessed 28 Feb. 2024.

Braiser Matt, Matheny Kevin. 2019. How to successfully implement API Management. Gartner. Available at: <https://www.gartner.com/en/documents/3913711>. Accessed 25 Jan. 2024.

Calder, Alan. 2005. IT Governance: Guidelines for Directors. Available at: <https://learning.oreilly.com/library/view/it-governance-guidelines/9781849281058/?sso_link=yes&sso_link_from=metropolia-university>. Accessed 18 Feb. 2024.

Calder, Alan; Moir, Steve. 2009. IT Governance: Implementing Frameworks and Standards for the Corporate Governance of IT. Available at: <https://learning.oreilly.com/library/view/it-governance-implementing/9781849281287/?sso_link=yes&sso_link_from=metropolia-university>. Accessed 28 Feb. 2024.

Clare, Pete; Powell, Lou. 2021. API Productization: The Three Essential Elements. Concentrix. Available at: <https://www.concentrix.com/insights/blog/elements-api-productization/>. Accessed 20 Feb. 2024.

De, Brajesh; Doda, Rajesh. 2017. API Management. Available at: <https://metropolia.finna.fi/Record/nelli15.3710000001118050?sid=406724569>. Accessed 4 Feb. 2024.

Hussain, Shakeer. 2023. What is API Developer Portal with Best Practices & Examples. Document360. Available at: <https://document360.com/blog/api-developer-portal-examples/>. Accessed 8 Feb. 2024.

IBM. 2023. Publishing an API. Available at: <https://www.ibm.com/docs/en/api-connect/10.0.1.x?topic=definitions-publishing-api> Accessed 6 Feb. 2024.

IBM. 2023. What is an Application Programming Interface (API). Available at: <https://www.ibm.com/topics/api>. Accessed: 6 Feb 2024.

Kong Inc. 2024. Types of APIs and Use Cases. Available at: <https://konghq.com/learning-center/api-management/different-api-types-and-use-cases>. Accessed 20 Feb 2024.

Kong Inc. 2022. What is a High Availability Cluster? Use Cases and Examples. Kong Inc. Available at: <https://konghq.com/learning-center/api-gateway/api-gateways-for-high-availability-clusters>. Accessed 10 Feb. 2024.

Oyova Software LLC. 2023. Do APIs Cost Money? How (And Why) To Charge for API Access. Available at: <https://www.oyova.com/blog/companies-charge-api-access-2/>. Accessed 6. Feb 2024.

Preibisch. Sascha. 2018. API Development. Available at: <https://metropolia.finna.fi/Record/nelli15.4100000007127464?sid=4075830758>. Accessed 6 Feb 2024.

Sindall. Gemma. 2023. What is API Governance? digitalML. Available at: <https://www.digitalml.com/api-governance-best-practices/>. Accessed 6 Feb 2024.

Smartbear.com. 2024. What is API Monitoring? Available at: <https://smartbear.com/learn/performance-monitoring/what-is-api-monitoring/>. Accessed 6 Feb 2024.

Software AG. 2022. API Management. Available at: <https://documentation.softwareag.com/webmethods/compendiums/v10-11/C_API_Management/index.html>. Accessed 20 Feb. 2024.

Thanh, Nam Le. 2023. What Is Backend Development? medium.com Available at: https://medium.com/@namtheartist95/what-is-backend-development-bcc6a15f8472>. Accessed 18 Feb. 2024.