DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Collaborative Development of Open APIs - Status Quo and Ideas for Improvements

Mihailo Rajacic

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Collaborative Development of Open APIs - Status Quo and Ideas for Improvements

Kollaborative Entwicklung offener APIs: Status Quo und Verbesserungsvorschläge

Author:	Mihailo Rajacic
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	M.Sc. Gloria Bondel
Submission Date:	May 15, 2020

I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, May 15, 2020 Mihailo Rajacic

Acknowledgments

I would first like to thank my advisor M.Sc. Gloria Bondel who provided me with the initial idea for the research topic and helped me a lot during all phases of this work. I would like to thank her for all the organized meeting sessions, valuable feedback, and the answered questions during the work on this thesis.

I also want to thank my supervisor Prof. Dr. Florian Matthes for providing me the opportunity to do this thesis for Software Engineering for Business Information Systems (sebis) chair at the TUM Department of Informatics. I would also like to thank him for the valuabe feedback and discussions, and for shaping the initial research topic idea into the thesis topic.

It is also important to recognize the big impact on this thesis from all the industry experts who took their time and accepted to participate in interviews for this thesis. Thank you!

Finally, I would like to express my gratitude to my family and all of my friends. Thank you for all the support you had for me during my studies, it was really important!

Abstract

Increased usage of Application Programming Interfaces (APIs) brought a lot of success to many companies that implemented and manage their API in such a way that it became a large source of their incomes. Still, a lot of issues exist around the APIs, and space for improvements exists in many areas of the API development.

The goal of this research is to analyze the current state and propose improvements to issues during API development processes related to collaboration. This is achieved through combined approaches of extensive literature review and semi-structured interviews. The literature review analyzes the current state of phases and roles involved in API development processes, as well as collaboration during these phases. Semi-structured interviews provide an insight into the most common challenges during API development from the perspective of experts in the area of APIs, as well as their views on the current and future state of collaboration between API providers and consumers.

The most important results of this research can be divided into a few different categories. Firstly, concept matrices for API development processes and roles provide an overview of more and less common process phases and API provider roles. Secondly, the collaboration between API providers and consumers is presented in different ways - through the analysis of current challenges in open API development, as well as through comparison between phases that should be collaborative in literature review and interviews. Finally, based on the previous research results seven different improvement ideas were proposed to improve API development processes, with the focus on collaboration. Unfortunately, the improvement ideas could not be validated because of time limitations, which leaves a space for future works.

Contents

1. Introduction

This chapter provides introduction to motivation and goals of this thesis, research questions and research approach, as well as outline of all chapters which are contained in this thesis.

1.1. Motivation

Application programming interfaces (APIs) have now already been around for decades, and first serious use cases where companies started using web APIs in their businesses were already seen in 2000, with Salesforce and eBay seen as pioneers (Anthony 2016). Today, we can find that in last five years an average of 2019 new APIs is added every year (Santos 2019) and the trend of growth has been happening for over a decade. Seeing such growths, it is visible some companies which provided APIs have made a huge business successes from their APIs, like eBay which generates 60 % of their income thorough APIs (Iyer and Subramaniam 2015) or Amazon that in 2016 had 33 open APIs with more than 300 diferent mashups - entirely new digital applications and services that integrate existing APIs (Evans and Basole 2016).

These positive trends and huge business opportunities are not equally distributed through different industry sectors, having the majority of successful APIs distributed among companies that are born-digital (Evans and Basole 2016), meaning that they belong to *"a generation of organizations founded after 1995, whose operating models and capabilities are based on exploiting internet-era information and digital technologies as a core competency*" (Panetta 2016), while it is very rare to see a company that is active in the API economy from industries like pharmaceutical or food industry (Evans and Basole 2016). These companies had to go through a process of digital transformation, and as a result there is an example Walmart in 2016 had only one API with only one mashup integrating their API. Still, they have made success in other areas of digital transformation like using customer data for personalizing the customer experience (Pamnani 2017).

Companies that have problems in capitalizing the potentials of APIs (Vukovic et al. 2016) can have them for a various reasons identified in business researches, but the recent research from (Smartbear 2019) showed that the most important technology challenge for the API is the need for standardization, described as *"the need to establish standards for how APIs are developed and maintained within the organization"* (Smartbear 2019). The same research (Smartbear 2019) showed that in case of quality or performance issues 34% consider switching to other API providers permanently.

1.2. Problem Statement

The above identified need for standardization shows the need to identify the current state of API development processes, their similarities and differences. That way, a holistic picture on API development processes could be seen, together with gaps that currently exist.

Another challenge that was identified was that many companies do not collaborate with external users and create provider-driven API that does not fulfill the consumer's needs (Bondel et al. 2019) from (Smith 2018). This challenge is not very common in the researches carried out until now and therefore needs to be additionally analyzed for its importance among experts in the field of open APIs.

Finally, standardization of APIs in terms of processes is incomplete if implemented only on processes, as together with business assets stakeholders are an integral part of API value chain (Glickenhouse and England 2016). It is also important to analyze the API provider side in details because API teams have closer relations to people using their product then other teams dealing with software development (Womack 2016).

1.3. Research Questions

In the following section four Research Questions (RQ) will be presented. These research questions are based on the challenges which were explained in the problem statement of this thesis and that will be used as basis for the further research.

RQ1: What are current challenges during Open API development?

This research question will be answered through the series of interviews with experts in the subject of Open API development lifecycle as Providers or during usage of Open API from API Consumer perspective. The main outcome from this research question is list of the most common challenges and if, and to which degree, lack of collaboration between providers and consumers of APIs is one of them.

RQ2: What are the phases that the development processes of Open APIs should contain?

This research question aims to analyze which are key phases during the development lifecycle processes of Open APIs with a focus on collaboration. To give a proper answer, information is collected through the literature review and compared through the comparison matrix. This matrix should show which phases are common for all identified processes and which are appearing only in some of them. Each of these identified phases is also checked for collaboration between API providers and API consumers in it. Analysis of key findings from RQ2 is done through series interviews with people experienced in Open API development or management.

RQ3: What stakeholders are involved in the collaborative development of Open APIs?

To answer this research question it is important to address it from different levels of detail. The first level is a high level where stakeholders are identified in their relation with the API value chain and second level where certain stakeholders are analyzed in a similar way as in RQ1. This means that a comparison matrix is formed between different roles that appear in different literature. In addition, for RQ3 analysis of key findings is also performed through a series of interviews.

RQ4: How could collaboration during Open API development be improved?

The goal of this research question is to create ideas for process improvements for the Collaborative Open API development. This research question will be answered based on the results from RQ2 and RQ3 from the literature review, as well as through RQ1 results and analysis of interviews with business partners.

1.4. Thesis Outline

The following section gives a short explanation of the thesis structure through its chapters, to provide the easier and more structured understanding for the reader.

Chapter 1: Introduction This chapter provides motivation and a goal for this thesis. Besides, research questions are presented and explained, as well as the research approach which was chosen to answer them.

Chapter 2: Foundations introduces the basis for this thesis from a theory perspective. In this chapter the technical definition of API together with API types is given and additional focus on managed APIs is explained. Also, the API value chain, API stakeholders, and collaboration in the context of APIs are defined. Finally, relations (similarities and differences) of API with Web Service, Service Oriented Architecture, and Service engineering are analyzed.

Chapter 3: Related Work This chapter presents an overview of scientific work related to this thesis. It includes works on collaboration, API lifecycles, and process analysis.

Chapter 4: Research Approach chapter presents an approach to the research used for answering the research questions. This includes both approaches for literature review and conducting the interviews with business partners.

Chapter 5: API Development Processes and Phases Analysis contains the presentation of extensive literature review research. It contains the matrix of different API development processes and phases they contain based on the literature review findings. It also contains the description of each of these phases found in the literature, as well as findings on the existence of collaboration in each of the phases together with explaining reasons for it, which is also based on literature review findings.

Chapter 6: API Team Roles and Responsibilities is based on similar principles stated in Chapter 5. It contains the literature review findings in the form of a matrix that matches different literature sources with identified API Team roles. Also, based on the research results each of the identified roles was matched with a proper explanation of responsibilities that go with it.

Chapter 7: Interviews Analysis presents the results of the analyzed interviews with experts. It contains the most important findings grouped into four categories. Each of these categories also has a further division into most important findings that were collected for it, having interviews that were used as sources mentioned.

Chapter 8: Discussion contains the similarities and differences between literature review and interviews findings on collaboration in different phases during Open API development lifecycles. It also presents the identified ideas for improvements in these processes, focused on improving the collaboration between API providers and consumers.

Chapter 9: Conclusion summarizes the work done during research through answers on research questions, addresses the limitations that were faced, and proposes future work.

2. Foundations

The following chapter provides theoretical foundation items for further research. The most important definitions, concepts, and terms relevant for this thesis are covered. These include: API definition from both technical and management side, API taxonomy, API Value Chain and Stakeholders, basic concepts of Collaboration in engineering, as well as relations of terms API and API development with other similar terms from the literature.

2.1. API Technical Definition and Types

In the following section a detailed API definition from a technical perspective as well as different views on types of APIs are presented.

2.1.1. API Technical Definition

Defining Application Programming Interfaces (APIs) requires having views from different perspectives in the literature, but before giving a proper definition we first need to focus on the need for APIs themselves. For the organizations, both born-digital and the ones that are digitally transformed today it becomes challenging to create customer-oriented solutions without ensuring reliable access to core information assets (Bell et al. 2018). Using APIs and the technologies and platforms that enable them these challenges can be addressed - APIs can act as doors to previously locked information assets and key business functionalities (Bell et al. 2018), and they can improve the user experience and satisfaction .

Looking at the technical perspective, in simple terms, APIs are a set of requirements that govern how two applications can communicate to each other. More formally, APIs "*define the contract of a software component in terms of the protocol, data format, and the endpoint for two computer applications to communicate with each other over a network*" (De 2017). Additionally, APIs are more precisely defined "*as a set of protocols, functions, mechanisms, tools, definitions, and attributes to share and develop new services across different domains and expand the existing services*" (Hussain et al. 2019).

Today, when we speak about APIs, we are likely speaking about Web APIs (RapidAPI 2019). Besides the Web APIs, precisely defined in the next paragraph, there are other APIs that do not need the network for their usage. A good example of such APIs is Windows APIs used by system hardware and applications like the API for communication between Microsoft PowerPoint and Microsoft Word (De 2017).

Web API refers to an API over the web which can be accessed using the HTTP protocol. It is important to emphasize that Web API is a concept and not technology because it should not be misunderstood with ASP.NET Web API Framework. It can be built using a wide range of technologies such as .NET, Java, and many others. Web APIs don't necessarily need to be RESTful. Web API implements protocol specification and incorporates concepts like caching, URIs, versioning, request/response headers, and various content formats in it (RapidAPI 2019).

In this thesis the focus will be on Web APIs, as they are the ones that can be productized and developed with the collaboration included.

2.1.2. Types of APIs

There are multiple classifications of which are the main API types in the literature. The basic division of API types is on private APIs and public APIs, based on the degree of visibility and access of APIs (De 2017). More detailed classification can be found in Table 2.1:

API Types	***Description***
Public APIs (De 2017); External APIs (Zhu et al. 2014)	These APIs are available both for developers which are internal to API providing company, as well external developers outside of the company. The interface of public APIs is designed to be accessible by a wider developer community for building software products like mobile and web applications. (De 2017)
Partner APIs (De 2017) (Zhu et al. 2014)	These APIs are specifically designed for partners to be able to access business functions in relation to the business partnership agreed between API producer and API consumer (Zhu et al. 2014).
Private APIs (De 2017), Internal APIs (Zhu et al. 2014)	These APIs are only exposed by internal systems and therefore less known and often meant for use inside the company. The company uses this type of API among the different internal teams to be able to improve its products and services (RapidAPI 2019).

Table 2.1.: API Types

There is a difference in the literature on the definition of Open APIs. While some authors take Open APIs and Public APIs as synonyms like (RapidAPI 2019) where they state ''*Open APIs: Also known as Public API, there are no restrictions to access these types of APIs because they are publicly availabl*e" others like (Moilanen 2017) state that ''*Public APIs are listed in catalogs and not hidden from anyone. This level includes open APIs and commercial APIs.*" Therefore, Open APIs in (Moilanen 2017) are defined as ''*APIs whose all features are public and which can be used without restricting terms and conditions*". During this research APIs which are publicly availabe were considered as Open APIs, even if they were monetized partially (freemium business model) or completely (e.g. pay per use or flat rate models). The most important criteria for one API to be considered as Open API was that it is offered to all interested developers.

2.2. API as a Product

In the section API as a Product the main goal is to make a clear distinction between API as a technical term and Product API, shown in the Figure 2.1.

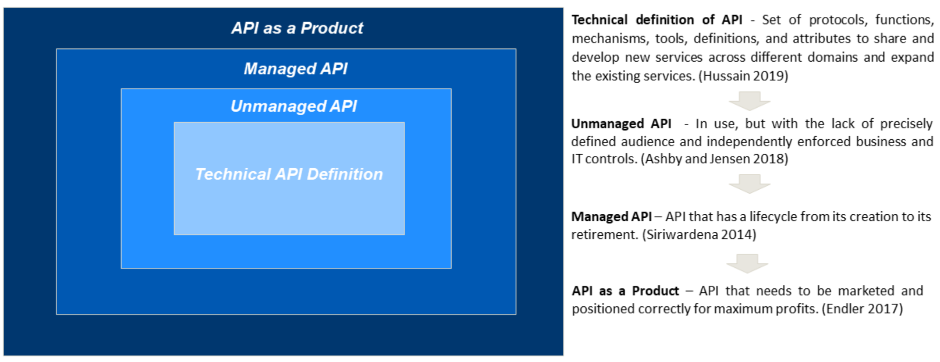
2.2.1. Managed APIs

Together with the technical definition of APIs another perspective needs to be considered in order to achieve a complete picture – management perspective. Unmanaged API and Managed API have few important differences like the lack of precisely defined audience and independently enforced business and IT controls (Ashby and Jensen 2018), but according to (Siriwardena 2014) another and maybe the most important key differentiator between an unmanaged API and a managed API is life-cycle management.

A managed API has a lifecycle from its creation to its retirement. In the different literature sources phases of API lifecycle vary. For example (Siriwardena 2014) shows that API lifecycle might contain Created, Published, Deprecated, and Retired stages. On the other hand, (Patni 2017) sees these phases as Analysis, Being Created / Development, Published / Operations, Deprecated and Retired, while (B. C. Doerrfeld et al. 2015) identifies Analysis, Development, Operations and Retirement stage.

2.2.2. API Taxonomy

After defining API and adding management layer to its definition, it is also important to answer the have a look at the different forms one API can take from the API Economy perspective begin with, in (Niinioja 2018) there is a detailed taxonomy of what an API can be, which can be seen in the Table 2.2. Looking at this table and seeing that Public APIs don't need to strictly be products, but also boundary resources or just service parts, opens another question: What exactly makes one API a product?



Technical definition of API - Set of protocols, functions, mechanisms, tools, definitions, and attributes to share and develop new services across different domains and expand the existing services. (Hussain 2019)

Unmanaged API - In use, but with the lack of precisely defined audience and independently enforced business and IT controls. (Ashby and Jensen 2018)

Managed API – API that has a lifecycle from its creation to its retirement. (Siriwardena 2014)

API as a Product – API that needs to be marketed and positioned correctly for maximum profits. (Endler 2017)

Figure 2.1.: API - From technical definition to API product

API Is...	*Description*	*Example*	*Types of API*
Important feature of a tangible product	API is a part of a tangible product or productized service. Customer gets the API as part of the deal when buying the product.	Internet of Things APIs for controlling state of things like home appliances or sensors	Partner API, Public API, rare Private API
Productized service	API in itself is a productized service, offered to all customers in the same way	Translation API, Payment API	Public API
Part of a digital or real-world service	API is part o the service experience, for example maintenance service is ordered with an API, or you can monitor package delivery with an API.	Logistics API	Partner API, Public API
Customer-specific service	API is part of a service offered to customers as a tailor-made solution including for example as integration to a service providers system	API is a customer specific application	Partner API
Interface to resources	API is just a means to access a resource the company is selling.	Company info APIs (risk category, owners, contact information)	Public API, Partner API
Interface to platform (boundary resource)	API is just a means to access a resource the company is selling.	Online Auction API, Apartment sharing API	Public API, Partner API
Part of integration	API is just a means to connect to applications and devices.	Company info APIs (risk category, owners, contact information)	Private API, Partner API

Table 2.2.: What can API be (Niinioja 2018) based on (Moilanen et al. 2018)

Looking only at their technical perspective, APIs can fit into the definition of technical boundary resources – "*resources that provide technical feasibility for the development of third-party applications and are used to access the platform*" (Myllärniemi et al. 2014). But as it was already mentioned, APIs are a lot more than their technical functionalities, and the reasons why they can be treated as products needs a look into the API economy, a discipline that works with opportunities to productize the exposure of business functions through APIs (Zhu et al. 2014).

If a certain organization considers API as a consumable product means that it needs to be marketed and positioned correctly for maximum profits (Zhu et al. 2014). In other words, an API becomes a product when it is managed like one (Endler 2017). This means that APIs like other software products have policies to govern their usage, ensure their care and feeding, and place limits on the amount of usage from a consumer (Holley et al. 2014). They also should allow consumption classified into different tiers, or monetization models (Holley et al. 2014). Summarizing the previous statements, APIs should be both products for the developers who build customer experiences through their applications and the mechanisms through which value is increasingly exchanged in modern economies (Google 2018).

2.2.3. API Value Chain and Stakeholders

Treating APIs as products and business values they bring requires an understanding of the API value chain. This is because according to (CA-Technologies 2015) understanding the API value chain will result in understanding the union of various business assets such as IT systems, internal and external personnel, client applications, and customers in efficient identification of the potential business value of those assets. API value chain elements vary in the literature, and combining the information from (Glickenhouse and England 2016), (Google 2018), (CA-Technologies 2015) and (Jacobson et al. 2012) we can group them into two sets which contain three elements described in the list below and represented in the Figure 2.2:

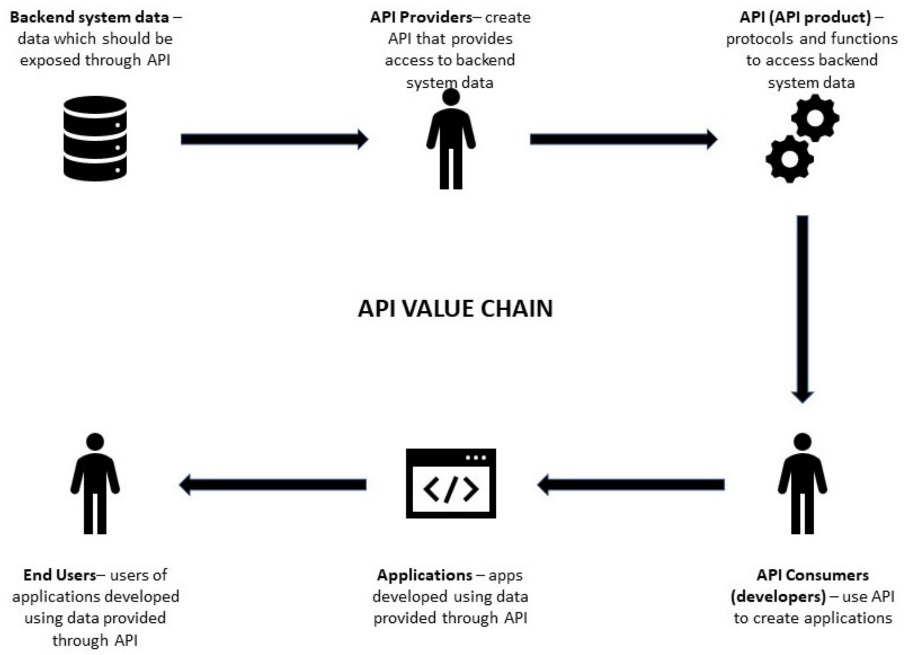

Backend system data – data which should be exposed through API
API Providers– create API that provides access to backend system data
API (API product) – protocols and functions to access backend system data
API VALUE CHAIN
End Users– users of applications developed using data provided through API
Applications – apps developed using data provided through API
API Consumers (developers) – use API to create applications

Figure 2.2.: API Value chain, based on (Queensland 2019)

Business Assets:

- Backend Systems Data – Data that should be exposed and used through APIs.
- API (API Product) – As explained in Sections 2.1 and 2.2.

API Value Chain Element	*Description Private API*	*Description Partner API*	*Description - Public API*
Backend System Data	Backend system data for which company may have no interest or right in sharing outside of their organization. (Jacobson et al. 2012).	Backend System Data for which company thinks it needs to stay inside a tightly controlled domain. (Jacobson et al. 2012)	Backend System Data for which company thinks it should be given to unlimited domain of developers.
API (API Product)	APIs that are only exposed by internal systems and therefore less known and often meant for use inside the company. The company uses this type of API among the different internal teams to be able to improve its products and services (RapidAPI 2019).	APIs that are specifically designed for partners to be able to access business functions in relation to the business partnership agreed between API producer and API consumer (Zhu et al. 2014).	APIs that are available both for developers which are internal to API providing company, as well external developers outside of the company. The interface of public APIs is designed to be accessible by a wider developer community for building software products like mobile and web applications. (De 2017).
Applications created using the API	Internal company-needed applications to automate certain business processes or help in different system integrations.	Applications which are developed for the work between partners, as well as applications for public use.	Applications which are developed for public or private use.
API Provider	API Team of a company which is the owner of backend system data or third-party company hired to build the API.	API Team of a company which is the owner of backend system data or third-party company hired to build the API.	API Team of a company which is the owner of backend system data or third-party company hired to build the API.
API Consumer	Application developers of the same company which owns and provides API.	Company's application developers and developers in companies (partners) that are allowed to access API.	Company's application developers and all other developers which want to build the app using provided API.
End User	Usually employees within company who uses an application built using API. External users also possible.	Anyone who uses an application built using API.	Anyone who uses an application built using API.

Table 2.3.: API value chain in different API Types

- Applications created using the API – Applications that will be provided to end users and that will use the data provided through APIs.

Stakeholders:

- API Providers – Team (roles further explained in the Chapter 6) which decides which data will be available through an API, establishes terms and conditions for usage, develops, deploys and manages APIs.
- API Consumers (Developers) – Stakeholders that use APIs to create apps under agreed conditions.
- End Users – Stakeholders which do not directly see or call APIs but use apps and hopefully benefit from that usage.

Even though there is a difference between Private, Partner and Public APIs explained in the Section 2.1 API Value chain remains unchanged. That is because whichever the chosen API Type is, all business assets and roles are still there, but with slightly changed descriptions, as we can see in the Table 2.3:

2.3. API and Related Terms

Web services, particularly Web APIs, are becoming the backbone of Web, cloud, mobile, and machine learning applications (Tan et al. 2016). Still, relations between different concepts and terms in this area are not well defined, with some of them being contradictory.

2.3.1. APIs and Web Services

To begin comparison between these two terms, we should first have a look at their definitions:

According to W3C definition from 2004 "*Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards."*

Coming back to API definition (Hussain et al. 2019), APIs are defined as "*a set of protocols, functions, mechanisms, tools, definitions, and attributes to share and develop new services across different domains and expand the existing services*".

What makes these two similar and what separates them? Speaking about similarities, from the above stated definitions, both are designed for system interaction, work under certain protocols and extend service functionalities. Since this thesis focuses on the Web APIs, can we then use the statement from (De 2017) that a web API can be considered as a subset of a web service? Other literature sources give an opposite opinion stating certain differences:

- APIs are protocol agnostic and they can use any protocol or design styles, while Web Services usually use SOAP. (Bush 2019)
- The goal of public and open APIs which are in focus of this thesis is to be transparent, offer open documentation and developer portals which should make onboarding of API

consumers easier. Web services are more closed and they offer only specific data and / or application functionalities to business partners. (Bush 2019)

Even though they are from the similar area of computer science and they work on the same principles, APIs and Web Services can't be seen as the synonyms and APIs can't be seen as subset of Web Services given their more open nature from protocol and management perspective.

2.3.2. API Development and Service Oriented Architecture

SOA (Service Oriented Architecture) is defined as software architecture based on loosely coupled software services that are integrated into a distributed computing system, by means of service-oriented programming. In SOA perspective, services are the building blocks of an enterprise. An enterprise is defined by its pool of services, both available internally and publicly, and the interaction pattern between the services(Suhardi et al. 2015). Taking this definition into the account, we will now see the relations between SOA and APIs. To begin with, let' have a look at similarities between SOA's main pillars and APIs. These include:

- Services must be reusable (Greca 2014) – Open APIs are public and their goal is to be reused multiple times by various applications.
- Services must have a service contract (Greca 2014)– APIs are contracts by technical definition.
- Services can be easily discovered (Greca 2014) – Ease of discovery of APIs is based on well written documentation and onboarding of API consumers
- Services can be composed into other services (Greca 2014) – Some APIs are provided in a way that behind the scenes compose services provided by other APIs.

Seeing the previous similarities, we raise the logical question : Does that mean APIs are just another name for services? Even though the list above presents many similarities between service and API, the most important difference between them that their designs do not share the same goal that should be achieved. (Mitchell 2018). According to (De 2017) there are few main differences between SOA and APIs:

- Even though APIs share the same technical characteristics as SOA, they are more open, developer centric and easily consumable. We saw the similar difference when comparing APIs to Web Services.
- APIs and SOA have different objectives behind them. (De 2017), (Mitchell 2018) and (Refalo 2016) agree that SOA helps in the agility and pace of the delivery of a service, while APIs drive the innovation, helping the pace of building apps.

In the end, seeing the similarities and differences, we draw the same conclusion as (De 2017), (Mitchell 2018) and (Refalo 2016): APIs and SOA are complementing each other rather than being contradictory.

2.3.3. API Development and Service Engineering

After reading the previous section, we should first make a clear distinction between SOA and Service Engineering. Service Engineering and Service Oriented Architecture stand on the different sides of disciplines related to services: Service engineering is a set of activities introducing a new business service, while SOA is a technical approach to redefine the enterprise business processes and form them as a group of IT enabled services (Suhardi et al. 2015).

Working definition for service engineering was taken from (Cardoso et al. 2008): *Service engineering can be referred to the set of activities involved in the development of service-based solutions in a systematic and disciplined way that span and take into account, business and technical perspectives. It is an approach that provides a discipline for using models and techniques to guide the understanding, structure, design, implementation, deployment, documentation, operation, maintenance and modification of electronic services.*

This definition of Service Engineering shares certain similarities with seeing Public APIs as product, where we can find that APIs like other software products have policies to govern their usage, ensure their care and feeding, and place limits on the amount of usage from a consumer (Holley et al. 2014) and that they also should allow consumption classified into different tiers, or monetization models (Holley et al. 2014).

Unfortunately no direct comparisons of these two terms were found during the literature review, so similarities and differences had to be identified directly comparing two definitions and differences between previously identified similar terms.

Similarities between API development and Service engineering stands on the fact that development of both Services and APIs should be performed taking into the account both technical and business side. Service engineering in its definition contains that it takes into account both technical and business perspectives, while treating APIs as products also needs to take business aspects besides APIs being technical by nature.

Different nature of services and APIs specified in the previous comparison between APIs and Web Services as well as with SOA, their development processes differ in a sense that APIs need to be developed with more agility and with more community involvement.

2.4. Collaboration

In previous chapters it was mentioned that there is a lack of collaboration during the development of Open APIs. While that stands, a new question may arise: What is actually collaboration in the context of this thesis? To begin answering that question, it is important to define the term collaboration in context of engineering: Collaboration in engineering is decision-making process that includes multiple stakeholders which resolve conflicts, seek for individual and collective advantages, agree upon courses of action and/or make joint outcomes which will serve their mutual interests (IJCE 2018).

API Development Lifecycle collaboration can be focused in two different directions – it can either be focused on internal integration of departments or external linkages with customers(Johnson and Filippini 2009). These two different types of collaboration are not mutually exclusive. Moreover, companies that are collaboration both internally and externally have the best chance to succeed (Johnson and Filippini 2009). The focus of this thesis is external collaboration and relationships with consumers of APIs.

Collaboration can be established in different levels of involvement, relations, and mediums and as a broad term it was formalized into a model that explains different levels and ways of collaboration. This model is called the 3C collaboration model and it was initially proposed by (Ellis et al. 1991), while other authors like (Borghoff and Schlichter 2000), researched the topic further and improved the model. For example, in (Ellis et al. 1991), collaboration was one of the 3c's together with communication and coordination, while in (Borghoff and Schlichter 2000) cooperation replaced it and collaboration was introduced as a term which stands one level above concepts covered in cooperation, coordination, and communication. These three elements of 3C collaboration model can be defined the following way:

- Communication – *„the interrelated behavior of two or more people and their interaction with the goal of transmitting information and understanding the content. "* (Bondel et al. 2019) based on (Leimeister 2014). In the context of API development this includes different kinds of promotions of APIs like developer portal where virtual communication happens and meetups, conferences or hackatons where providers and consumers can meet face to face (B. C. Doerrfeld et al. 2015). This will help consumers gather the important information and understand the content (API) better.

- Coordination – *„the matching of decentralized actions and decisions of interdependent organizational units on the basis of suitable communication processes with regard to the optimal fulfillment of the goals. "*(Bondel et al. 2019) based on (Leimeister 2014). Examples of coordination during API development are preparations for changes during API lifecycle, which include new versions or complete deprecations of existing APIs (B. C. Doerrfeld et al. 2015). Such changes need to be done in coordination in order to avoid bad reputation for API (provider interest) as well as to keep applications built using that API running (consumer interest), but without actions done together.

- Cooperation – *„the activity of two or more individuals, which is consciously planned and coordinated with one another to ensure the achievement of the goals of each individual involved to the same extent "*(Bondel et al. 2019) based on (Leimeister 2014). Cooperation is the highest level of customer involvement and for API development it can be represented in including consumers during Strategy, Design or Mockups. For example, this can be represented in (IBM 2016) where consumers are provided with mockups so they can test the API and provide the valuable feedback which can be used for further building. This approach will fulfill the requirement that goals of each individual are achieved as the API will be tailored to consumer needs.

Seeing the previous 3C model elements and explanations it is clear that in the focus of this thesis all of these elements are considered as parts of collaboration between API Provider and API Consumer.

3. Related Work

During this research one of the findings was that there is a very little academic research in the area of API Management and industry researches and experiences provide a lot more insight into this topic (Fremantle et al. 2016). Research on API development processes, collaboration during API development process or stakeholders involved was therefore even less. Nevertheless, certain key concepts related to the work done in this thesis were covered in the past in academic and business research. These include:

Challenges during API Development Process - this research topic was covered in scientific and business articles and it was important as the motivation to begin working on this research. From the scientific articles (Evans and Basole 2016) worked on the analysis of APIs provided and applications built in different business sectors. These research results gave clear distinction in success of APIs between companies which are digital and others which are in other sectors.Many authors also worked on challenges in specific during the API development, like (Vukovic et al. 2016) that focuses on API strategy or (Uddin and Robillard 2015) where challenges during API documentation writing were analyzed. On the other hand, studies like (Smartbear 2019) or (Smith 2018) did the business research in which interviews various business experts were conducted and certain issues were identified. These included: User (Consumer) Experience, Security or Lack of Standards.

Improvements if the field of Open APIs - there are various literature sources which cover the topic of improving APIs from various perspectives - from completely technical aspects to management aspects. For example, (Costa 2018) concentrates only on improving API performances through caching. On the other hand, (Myers and Stylos 2016) improves design and documentation of APIs working with stakeholders (user-centered), combining both technical and management side. Finally, (B. Doerrfeld et al. 2016) writes on improving the Marketing of APIs, which only concentrates on management perspective of API development process. Unfortunately, none of the literature sources found analyzes improvements from the perspective of complete API development process.

Collaboration - there is a reasonable number of scientific papers and books which cover the topic of Collaboration during product development, most of them from business and management perspective. These include different collaboration models and types of collaboration like 3C model from (Ellis et al. 1991) and (Borghoff and Schlichter 2000). The number of papers and books which cover collaboration during API development is on contrary very limited. Examples include paper from (Bondel et al. 2019) collaborative API management tool was designed and certain aspects of collaboration during API development processes were analyzed.

API Management - unlike other relevant topics mentioned until now, API management is well covered in books and business articles / brochures. Many authors like (De 2017) or (Patni 2017) provide overview in all phases of API development lifecycle management with guidelines for their successful implementation. Also, a large number of companies which work in different aspects of API management like (B. C. Doerrfeld et al. 2015) or (Akana 2015) provide very detailed sources of information on this topic.

Process Analysis - for this research topic related work contains concepts from the paper (Beverungen et al. 2018). This paper covers some similar approaches to this thesis in the field of Service engineering, defined in Foundations chapter. These similar approaches include identification of different concepts with their detailed descriptions, as well as analysis of different processes for Service engineering compared in a concept matrix.

4. Research Approach

4.1. Literature Review

The research was carried out as an extensive literature review based on (Watson and Webster 2002) principles and a matrix that includes different concepts was identified. During the research, search through most common terms was used, and the majority of API Processes was identified in business-related articles, brochures, and books. Search in scientific web sites and search engines were less successful. One of the main reasons for that was already mentioned in the Related Work chapter - the area of Web API Management lacks the precise definitions and there is very little academic research done in this area (Fremantle et al. 2016).

Two libraries and their search engines were mainly used during this research - OPAC University library of the Technical University of Munich that contains library's search engine and provides access to books and articles from various other sources like ResearchGate, Springer Link, IEEE, etc. and Google Scholar. As explained above, search through the most common terms was used in both libraries. Most important from these terms were:

- **For API Processes analysis**: API Process, API Lifecycle, API Management, API Phases, API Development, API Governance.
- **For API Roles and Stakeholders**: API Team, API Roles, API Stakeholders, API Provider, API Consumer.

Search results were not satisfying as through all these searched terms there were only two API processes found, none of which came from the scientific papers but as chapters of books on API management. Since this number was not good enough to create the comparison matrix, the next phase in research needed to be open web searching, using the same most important terms previously mentioned. After applying both research in academic libraries and open web research, a total number of 13 API Processes and 7 API Roles and Stakeholders divisions were identified. These vary in their:

- **Types of sources** – While the majority of identified processes and roles come from books, there is also a certain number taken from business web articles as well as from the shorter brochures which document a certain API Management Platform.
- **Levels of details** – Some of the identified processes have very high-level of identified phases like (De 2017) where only 5 phases were provided (API Proposal, Technical requirements gathering, Building and validation, General availability, Adoption and Sunsetting), other like (Google 2018) provide more phases (Design, Develop, Secure,

Publish, Scale, Monitor, Analyze, Monetarize), but there are also processes which combine the previous approaches like (Jacobson et al. 2012) and provide phases (Analysis, Development, Operations, Retirement) which are split into sub-phases (e.g. Analysis phase is split into API Business Strategy, Monetization, Understand Target API Customer). Descriptions of these phases also varied in the literature, from only a few sentences to complete book chapters.

- **Naming conventions** - Different naming conventions also lead to the fact that phases with the same names may have different meanings and vice versa. For example, API Strategy phase in (Massé 2019) has a similar description and as Plan phase in (Broadcom 2017), so both of them were classified under the same API Strategy phase, but since the description of (Broadcom 2017) also included prototyping, Plan phase from this source was also classified under Mock / Simulate phase. On the other hand, the Design phase in (Google 2018) has much fewer details than (Vester 2017) where it includes sub-phases and it could be classified into more than just the Design phase.

Each of the identified processes was standardized into the list which included API lifecycle process phases and most important details on each of these phases. Based on these phases and descriptions processes were analyzed and classified. In the end, API processes analysis classified different phases in API lifecycle into 14 major phases.

A similar approach was used for API roles and responsibilities. All of the identified roles were standardized into the list, together with their responsibilities which were written as a description to each of the identified roles. During the research 13 different roles were identified and matched with the literature sources where they were identified.

4.2. Interview Design and Analysis

In the second phase of this research the goal was to analyze challenges and collaboration during API development from the perspective of experts who work with them, from both API providers and API consumers. To achieve these goals the series of semi-structured interviews was conducted, based on the principals defined in (Adams 2015).

The approach with using semi-structured interviews was chosen as it was the best fit for the research needs. It allowed between standardized, mostly closed-ended surveys and free form, open-ended sessions (Adams 2015). It allowed avoiding the limited scope of question answers which is usually a result of closed-ended surveys, but also a certain degree of structure and time limits which was also significant knowing that these interviews should fit in schedules of interviewed experts.

Participants for these interviews were selected from the experts in various API related topics. One of the main criteria during the selection of participants was to achieve diversity in perspectives of these experts and analyze differences in them, especially between API providers and API consumers. Participants in these interviews included software developers who developed applications using Open APIs, API Engineers as well as experts from the field of API Management. They also varied in their experience, having the least experienced participant with 2.5 years in the field of APIs, while the most experienced participant works in this field for more than 30 years. Table 4.1 contains a detailed overview of interviewees which participated in these interviews. Interview names coding was used to summarize the analysis results in a more understandable way.

Interviewee ID	*Role*	*Industry*	*Experience*	*Provider / Consumer*
I1	Software Developer	Education	3 years	API Consumer
I2	Software Developer	Legal	9 years	API Consumer
I3	Developer Experience Director	E-Commerce	20 years	API Provider
I4	API Engineer	Video Streaming	10 years	API Provider
I5	Software Developer	Automotive Industry	2.5 years	API Consumer, API Provider
I6	Director of Applied Technology Research	Business Information Systems	30 years	API Consumer
I7	Software Developer	E-Commerce	3 years	API Consumer
I8	Integration Architect	Fashion Retail	12 years	API Provider
I9	Product Manager	API Tooling	4 years	API Provider
I10	IT Software Industry	Software Developer	5 years	API Consumer
I11	IT Software Industry	Technical Lead	8 years	API Provider

Table 4.1.: Interview Participants

The interview questionnaire followed the guidance from (Adams 2015) in almost all of the given guides with the exception that it didn't contain any closed-ended questions. Interview questions were edited and polished during meetings between student and advisor, and one pilot interview was conducted to test the questions structure, length, and online interview format. In the end, the questionnaire with three sections was established.

In the first section, participants were introduced to the topic of research and formal interview information such as recording permission and anonymization of results. The second part of the questionnaire contained four questions about the interviewee and his / her company, including industry, position, and experience information. Finally, in the third section questionnaire was focused on the research related questions. It consisted of five open-ended questions regarding challenges during API development, collaboration during the API development lifecycle, analysis of the API development lifecycle phases identified during the literature review, and finally about API provider roles that are involved in the collaboration. The questionnaire was uniform in most of its parts, but it also contained certain questions that were tailored depending on the interview being done with API provider or API consumer.

The complete questionnaire can be found in Appendix A of this thesis.

During this phase a total number of 11 interviews was carried out. All of the planned interviews were carried out online through various communication platforms such as Skype or Zoom, as given the pandemic situation in March and April 2020 face to face interviews were infeasible. Interviews were planned to be between 30 and 45 minutes long. In average, interviews lasted 33 minutes and 45 seconds, which meant that the initial length goal was fulfilled. All of the interviewed participants accepted that the interviews are recorded and transcribed, which made further qualitative analysis easier to perform. Transcription was done by transferring the audio original interviews into the written format and cleansed from the transfer errors using the method of 'corrective listening' - meaning that grammar and speech mistakes were corrected (Schmidt 2004)

Qualitative analysis was done following the principles from (Schmidt 2004).This type of analysis was found suitable for the analysis knowing that the technique of open-ended questions (Schmidt 2004) was used during interviews. Five different stages for analyzing the collected data were presented in this paper and therefore used during the analysis. These stages are:

- **Material-oriented formation of analytical categories** - In this stage material in the form of transcribed interviews was read and analyzed for the formation of analytical categories. It included a lot of connection to already identified concepts in research questions, literature review and questions which were asked. It was important though to find out some new concepts and categories which appeared in interviews but which were previously not identified.
- **Assembly of the analytical categories into a guide for coding** - Here the most important part was to finalize the initial categories formed in the first stage, and also to once again think about them and possibly change them based on various criteria like theoretical assumptions, field experience, or material collected (Schmidt 2004).
- **Coding of the material** - Using the results from first two steps, established analytical categories are applied to the material and the material is classified into important categories, around which certain parts of analyzed material tend to group (Schmidt 2004).
- **Quantifying surveys of material** - Quantifying in the context of qualitative analysis doesn't necessarily mean forming a lot of graphical and statistical categories, but rather counting the frequencies of identified terms and concepts in interviews. The concept of structuring the identified numbers in tables (Schmidt 2004) was not entirely followed, but the identified numbers were included in lists when cases were interpreted in the fifth step of this analysis.
- **Detailed case interpretations** - Results of material coding were interpreted through the list of identified concepts and opinions on these concepts in the analyzed material.

Outcomes of the work defined in this section are presented in the Chapter 7 - Interviews Analysis.

5. API Development Processes and Phases Analysis

API Processes analysis research results are represented through two tables – Table 5.1 which represents all identified API processes which were identified and comparison of phases which each of them includes. If the API process contains a certain phase, it is marked with '✓' character. Table 5.2 represents the existence of collaboration in each phase. Examples were written in phases where collaboration exists, combining different descriptions from the identified API processes.

5.1. API Development Processes and Phases - Literature Review

The following list presents the 14 different phases during found in API development processes and their descriptions. All of these descriptions were formed using descriptions identified in the processes found during literature review. These processes are represented in the Table 5.1. It is important to mention that the following phases don't represent the strict sequence of events from API Strategy until the Retirement plan, and that API building should not be taken as a project with fixed start and end date (Google 2018). Modern APIs are treated as products with multi-iterative lifecycle (B. C. Doerrfeld et al. 2015). This means that API Product should be developed in accordance with developers' preferences and needs, and should provide increasingly more useful and intuitive experience for them (Google 2018):

- **API Strategy** - API strategy phase should create business plan for an API in entirety. This phase should be performed even before any design or development starts. In this API phase providers should gauge interest from the targeted audience, perform market research, forecast trends within the sector, and make API usage projections (B. C. Doerrfeld et al. 2015). This strategy should be aligned with corporate API strategy.

- **Design** - API design begins with the protocol selection (REST, SOAP, GraphQL), which is followed by writing user stories in order to understand the customers better (TIBCO 2019). According to (Arnny 2017) next steps after that are creating the logical data model and translation into logical service / API groupings.

- **Mock / Simulate** - Mock / Simulate phase in API development process includes modelling of API resources, API operations and methods, and API request/ response payloads and codes (TIBCO 2019). The actual API itself is not being built, but it is important to get feedback from stakeholders or product in this phase and to review and analyze the results (Vester 2017).

		Phase													
Source	Source Type	API Strategy	Design	Mock / Simulate	Build	Test	Secure	Publish	Document	Onboard and Engage	Marketing	Version	Monitor and Analyze	Monetize	Retirement Plan
(Bell et al. 2018)	Book		✓		✓	✓	✓	✓				✓	✓		
(Broadcom 2017)	Web Article	✓	✓	✓	✓	✓	✓	✓		✓		✓	✓		
(Google 2018)	Book		✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	
(Massé 2019)	Web Article	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓
(De 2017)	Book	✓	✓		✓	✓	✓	✓		✓	✓	✓	✓	✓	
(B. C. Doerrfeld et al. 2015)	Book	✓	✓	✓	✓	✓	✓	✓			✓	✓	✓	✓	✓
(Axway 2019)	Web Article	✓	✓		✓	✓	✓	✓		✓	✓	✓	✓	✓	✓
(Patni 2017)	Book		✓	✓	✓	✓	✓	✓		✓		✓	✓	✓	✓
(Vester 2017)	Brochure		✓	✓	✓	✓	✓	✓				✓	✓		✓
(Akana 2015)	Book	✓	✓		✓	✓	✓	✓	✓	✓		✓	✓	✓	
(Arnny 2017)	Brochure		✓	✓	✓	✓	✓	✓		✓		✓	✓		
(TIBCO 2019)	Brochure		✓		✓	✓	✓	✓	✓			✓	✓		
(TransparentData 2019)	Web Article		✓		✓	✓	✓	✓				✓	✓		

Table 5.1.: API development processes and phases - Concept matrix

- **Build** - In build phase API is actually developed with the complete functionalities. Build phase should confirm the from Design phase (Patni 2017). API needs to be built in accordance with principle ″fail fast, succeed faster″ having best practices in mind (Arnny 2017) and should be fully integrated into back-end system as well as API Portfolio (Patni 2017).

- **Test** - Testing is an important phase as it is required to monitor and fix potential issues with the API before any consumer faces the problems (B. C. Doerrfeld et al. 2015). In this phase we should ensure that newly created API meets performance, functionality and security expectations (Axway 2019). These tests include performance-oriented tests, functional tests and use case tests (B. C. Doerrfeld et al. 2015).

- **Secure** - In secure phase we should protect API from threats and vulnerabilities (Broadcom 2017). This includes adding options for thresholds and varying service levels and setting access levels to the API (Vester 2017).

- **Publish** - Publishing of the API does not require a lot of work, but it is still a big milestone for the API development (Patni 2017). Publishing is the process of making

APIs available to app developers for consumption. (Google, 2017) After this step the right communities can see and access the data available through the API (Bell et al. 2018).

- **Document** - Even though every API needs to be documented, only few of processes found identify documentation as a separate step. In these processes document step is defined as well-formatted, searchable document which should help developers understand the API quickly, identify and address issues they may find (TIBCO 2019).
- **Onboard and Engage** - Onboarding and engage phase should help developers understand key API capabilities (Axway 2019). Besides access and communication through blogs, personal contact and trainings, it also includes recognition of the developers who use API in new and innovative ways and involving developers in API changes which come in other phases (Akana 2015).
- **Marketing the API** - To market the API, it needs to be ran as a product. API marketing, just other kinds of marketing offers wide variety of promotions. First and very important is developer portal, which provides access to important API resources. API then needs to be 'evangelized' – promoted through online media, conferences, meetups or hackatons. Finally, API needs to be well supported in terms of quality and transparency (B. C. Doerrfeld et al. 2015).
- **Version** - This phase is not simply about publishing a new feature or complete version of the API, but also about maintainance of APIs in order to ensure that they are always up to date and the integrity of APIs isn't endangered (Axway 2019). It also needs to ensure that all more or less significant changes on the API are done with consumers and internal systems in mind. Changes need to be compatible with previous versions and downtime needs to be minimal (B. C. Doerrfeld et al. 2015).
- **Monitor and Analyze** - Monitoring and analysis in terms of APIs needs to be done properly in order to understand the need for further development and potential changes (B. C. Doerrfeld et al. 2015). It should allow tracking API usage, load, transaction logs, historical data and other metrics that can indicate usage status as the success of the API (Arnny 2017).
- **Monetize** - Monetization is API development phase which is considered in different times during API development process. Some processes consider it in the beginning, during the design and strategy, while others consider it after publishing. Monetization of APIs is a choice is between offering APIs for free when there is an opportunity for it or monetizing APIs and drive additional revenue for the business (Axway 2019). There are various models for API monetization like charging developers or paying them through revenue sharing (Google 2018).
- **Retirement Plan** - Creating a good retirement plan starts with identifying a proper time and reasons for retirement(B. C. Doerrfeld et al. 2015). Retirement might happen for

different reasons including technology changes and security concerns (Axway 2019). Once identified, retirement needs to be properly done and communicated to avoid negative reactions among consumers (B. C. Doerrfeld et al. 2015).

Results of this literature review showed that all of the identified processes have the technical phases such as Build or Secure, but where they start do differ is on management-related phases like API Strategy, Marketing the API, or Monetization. This shows that the API as a product approach still has a room for improvements and that the standardization issue stated in the motivation shows its signs here.

5.2. API Development Processes and Phases - Collaboration

In the following section previously identified, API development processes are analyzed for the existence of collaboration. Presented in section 2.4 of this thesis, collaboration can involve consumers on different levels, they can either communicate, coordinate, or cooperate with the providers. Table 5.2 presents the phases based on collaboration that is marked with Yes / No, and for each phase an example from the literature is presented to illustrate it in more detail. Collaboration was in the end found in 8 out of 14 identified phases, and most of the identified phases that were collaborative were more management oriented.

Phase	*Collaborative?*	*Example*
API Strategy	Yes	API Strategy requires talking to different stakeholders such as competitors, regulators, partners or independent developers (De 2017).
Design	Yes	Some of the identified processes like (Vester 2017) include feedback as sub-phase of Design phase. The purpose of this feedback is to bring stakeholders to analyze the initial results (usually mocked) and compare them with the expectations set during initial design.
Mock / Simulate	Yes	In early phases of API development providers should build models and mockups which allow collaboration with developers (consumers) to confirm and ensure that API is going in the right direction (TIBCO 2019), or to identify the problematic areas.
Build	No	
Test	No	
Secure	No	
Publish	No	
Document	No	
Onboard and Engage	Yes	Engage phase is almost entirely dedicated to communication with developers and making them understand the key capabilities of the API (Axway 2019) and guide them to use it in the proper way (Massé 2019)
Marketing the API	Yes	Similar to engage phase, this phase is also by nature collaborative with external stakeholders, as it includes various promotions of the API like online media, conferences, meetups or hackathons (B. C. Doerrfeld et al. 2015). Most of these promotion ways will virtually or physically bring consumers and allow collaboration with them.
Version	Yes	As it was already mentioned when Version phase ws explained, changes on the API are done with consumers in mind. Changes need to be compatible with previous versions and downtime needs to be minimal, which includes coordinating all movements during version phase with customers. (B. C. Doerrfeld et al. 2015).
Monitor and Analyze	No	
Monetize	Yes	It is important that the monetization for the API brings values to both developers (consumers) and end users, as well as to create financial gain to API provider (B. C. Doerrfeld et al. 2015). This means that 'voice of the consumers' needs to be listened and some form of collaboration (at least communication) needs to be established in order to make monezitation work.
Retirement Plan	Yes	As already stated in the previous section, once when decision about retiring the version or the complete API, it needs to be properly done and communicated to avoid negative reactions among consumers (B. C. Doerrfeld et al. 2015).

Table 5.2.: Collaboration in API Development process phases

6. API Roles and Responsibilities

In the section 2.2.3 we have mentioned API stakeholders as key members of API Value chain. While roles of API Consumer and API End User do not have any further classifications, it is different with API providers, where literature offers various divisions of API provider team roles and responsibilities in order to achieve a successful fulfilling of different phases during API development. API teams can be as small as containing a single member responsible for both implementation and operations to multiple teams which cover engineering, operations, product, support, community, and management (Bortenschlager and Willmott 2017).

API team is similar to other software development teams, with a few key differences, like being closer to the people using their product because they are usually business partners (Womack 2016). These differences can lead to creating new positions within current organizational structure or to assigning additional responsibility to existing roles (Akana 2015). Table 6.1 offers comparison matrix of different roles identified in the literature, while text after that explains these roles further including their responsibilities.

			Role												
		Source Type	API Product Manager	API Architect	API Developer	API Evangelist	API Champion	Service Manager	Technical Writer	API Support/Operations	Quality Assurance	Sercurity Architect	Marketing / Branding	Analyst	Legal
Source	(Bell et al. 2018)	Book	✓	✓	✓				✓		✓				
	(Akana 2015)	Book	✓		✓	✓			✓	✓					
	(Google 2018)	Book	✓	✓	✓	✓	✓								
	(Zhu et al. 2014)	Web Article	✓		✓						✓				
	(Preibisch 2018)	Book	✓	✓	✓				✓	✓	✓	✓			
	(Jacobson et al. 2012)	Book	✓		✓	✓					✓		✓		✓
	(Womack 2016)	Web Article	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	

Table 6.1.: API Team - Roles in different literature sources

Roles identified in the table above have the following responsibilities:

- **API Product Manager** - role responsible for turning API plan into the action by looking after each stage of the API lifecycle. Different sources include different range of activities for API product manager, including involvement in design, implementation and versioning (Bell et al. 2018), billing for API usage, branding and marketing (Akana 2015), to performance evaluation and monitoring (Zhu et al. 2014). Some of these activities are in other sources assigned to different roles, while API product manager is focused on strategic and higher-level activities.

- **API Architect** - mostly included in early phases of API development like strategy and design, but they also have the responsibility to be involved in development and testing and ensure that developed products meet the agreed standards (Google 2018). They also need to be involved in design and integration of any updates and new versions, as their role is to understand the problem and help creating the technical solution (Womack 2016).

- **API Developer** - represents the team member who does the actual building of the API. They should produce APIs that are easy to use and highly consumable (Google 2018) and in line with the given architecture and security guidelines (Preibisch 2018). Some of their responsibilities may as well overlap with other comparing the different sources: writing documentation (Google 2018), testing (Zhu et al. 2014), supporting users (Jacobson et al. 2012).

- **API Evangelist** - also called community manager, developer evangelist or developer relations is one of the most important roles in API team. This role should not be the same as API Operations support, meaning solving technical issues with APIs, but it should rather have needs of developers and what they need to be successful (Google 2018). In other words, API evangelist should ensure that the developers derive value from the APIs and that they are values by API Providing company (Akana 2015).

- **API Champion** - one of the roles which is often covered by other existing API team roles, mostly by API Product Manager. When existing, this role has the responsibility to convert technical metrics into values that are understandable to the business executives and also ensure stable financing of API projects (Google 2018).

- **Service manager** - this role appears only in the book (Bell et al. 2018). This book explains service role as the person responsible for managing resources which define backend services.

- **Technical Writer** - technical writer should write the documentation in accurate, complete and accessible way, as well as to ensure that the documentation follows the most recent version of the API (Akana 2015). According to (Womack 2016) another responsibility of this role is to manage the blog and instructions on the API usage.

- **API Support / Operations** - different literature sources considered this role as either separated into two different roles API Support and API operations like in the book (Akana 2015), but in this thesis we will use the classification from (Womack 2016) and consider them as one role, because they are complementing. API support role is responsible for ensuring the smooth running of the API by solving technical issues of API Consumers (Akana 2015), while Operations should be involved in API deployment and technical management of APIs by monitoring uptime and performance (Womack 2016), (Akana 2015), (Zhu et al. 2014).

- **Quality Assurance** - quality assurance role has to be done during API development but in the majority of identified literature there is no specific role which should do this job, but it is rather assigned to other existing roles. When there, this role should ensure that API is delivering the output that is expected (Jacobson et al. 2012).

- **Security Architects** - role which does not always exist in API teams, as usually API developers take security related responsibilities (Preibisch 2018). If companies insist on security by design (Preibisch 2018) then this role should make sure that design follows security standards of the web and the organization (Womack 2016).

- **Marketing / Branding** - role which is connected with API as a product principle. This role has various responsibilities in promoting the API, from working on different promotions on developer website to organizing events and creating visual campaigns (flyers, banners, stickers etc.) (Womack 2016)

- **Analyst** - member of the API team that should be connected with all other roles in API team. It should work with others on potential issues and improvements of the API. These issues and improvements should be based on analysis of data sets, support tickets, market trends, customer patterns and other (Womack 2016).

- **Legal-** this role in API team establishes rules and guidelines for using the API. It needs to assure that corporate and customer data is being used in an appropriate way and that content licensed from third parties is in line with regulations (Jacobson et al. 2012).

This analysis shows that there is a large number of roles that are not common for most of the identified roles and responsibilities divisions found in the literature. Also, there is a large number of roles that can be substituted with other roles, like quality assurance often being done by other team members, or security architecture often falling in the hands of developers. Chapter 7 presents which of the roles could be involved in the collaboration between API providers and consumers on the provider side, based on the experiences and views of interviewed experts.

7. Interviews Analysis

The Interviews Analysis chapter presents the most important findings of the qualitative analysis based on 11 interviews done during this research. It is the outcome of the analysis based on the principals presented in section 4.2 of this thesis.

The most important findings were grouped into four analytical categories, and they will be presented on the following pages. For each of the identified categories, the most important findings are grouped and assigned to interviews where they were discovered using the interview codes. All descriptions that explain the identified findings come as a summary of information collected during interviews.

7.1. Analysis - Identified categories

The following section presents analysis results grouped into four different analytical categories:

7.1.1. Identified challenges during Open API development lifecycle

This analysis category came out of the specifically asked question during interviews, where interviewees were asked to identify challenges during development of Open and Private APIs. It was important do address this question for another reason - to get the answer on RQ1 - *'What are the current challenges during Open API development?'*. It was also interesting to see to which level the interviewed experts see topics related to collaboration as an issue. In the following list the identified challenges which exist during open API development, in comparison with Private APIs, are presented:

- **Lack of knowledge about consumers and their requirements** - Mentioned and explained by the largest group of interviewees (I1, I3, I5, I6, I8, I9, I11), this challenge makes one of the biggest differences between developing an Open API and the one which is private on the other hand. It means that once consumers are 'on the other side of the table' with a clear set of requirements and usually the same viewpoint on the problem which comes from the same object domain. It becomes a lot different with Open APIs, where a large number of consumers can use the API and therefore it becomes almost impossible to know all of their potential use cases and viewpoints. This in the end brings a lot more work on the side of providers.

- **Lack of consumer involvement in the design** - This challenge, identified by two interviewees (I3, I5) is directly related to collaboration. For the interviewees who identified this issue this challenge came out as a consequence of the first challenge identified in this list, lack of knowledge about consumers and their requirements. It brings a consequence that on the contrary from Private APIs where design can be agreed upon and verified directly with consumers in every stage for Open APIs providers usually go with 'the best guess' approach in design, and on their side usually create a design based on what they think their customers would like to use.

- **Lack of mechanism for systematic feedback** - Identified by one interviewee (I1) from the API consumers' side, this issue is defined as a lack of developed process for collecting the feedback from the wider range of consumers throughout the whole API development lifecycle. It also means that API consumers are not encouraged to give any kind of feedback, as they don't explicitly see their benefit from it. It is different with private APIs as such mechanisms don't need to exist, knowing the reasons explained in the first two issues.

- **User experience should be the priority** - Also identified only once (I9), this challenge comes from the business side, as for the Open APIs there is a lot of competition in the market, and if the experience for users is not on the expected level, consumers will easily switch to a different API product. In the example of Private APIs, it is completely different as the targeted user is known and API is specifically built for them.

- **Change management** - Management of changes, the challenge which appeared twice (I4, I8), from minor changes like field updates to more significant ones like versions, require a lot of carefully made decisions as API can't be down because of a huge number of users and interactions of one API into their applications. For private APIs this is not the case, as users are known and they can be directly contacted and informed in case of every possible need for maintenance when API can possibly be in its downtime.

- **Security and privacy** - An entirely technical challenge, discovered in interviews with a large number of experts (I1, I2, I6, I7, I9, I10, I11). This challenge provided different points of view from different interviewees, some of them clearly stated difference in security levels needed as something that is still a challenge which has to be solved, others confirmed the challenge stating that it is a topic from the past which is now easily solved using API Management Platforms. Finally, there was a third opinion that there is a completely equal level of security and privacy needed for private and open APIs.

- **Monitoring and analytics** - In this challenge, mentioned in interviews (I1, I2, I4) the focus was once again on the technical side of API development and differences in open and private APIs in this phase. It means that monitoring on numbers of API calls and other parameters of API usage needs to be monitored having the undefined number of potential users and the ways they use it. It is not the case with private APIs where a limited group of internal users can not create this kind of issue.

- **Documentation** - Issues with documentation were explained in three different interviews (I2, I6, I7, I10, I11), and they state that for different purposes and audiences that open and private APIs have there is a need for different documentation types. This means that open APIs need to have much more solid documentation with a lot of details, as many consumers will have this documentation as to their first contact with the API and this documentation should not be only a view from a provider perspective, and that documentation also lacks consumer involvement in its creation and update processes.

- **Legal issues** - Challenges in the legal domain mostly came from interviewees who have experiences with new legal regulations such as GDPR (I2, I6). Data provided through open API needs to be checked for GDPR or other legal domain regulations, which is not the case with private APIs where data doesn't go out of the company domain.

Seeing the identified issues it is visible that a reasonable number of them can be connected to lack of collaboration between API providers and consumers. These are: Lack of knowledge about consumers and their requirements, lack of consumer involvement in the design, lack of mechanism for systematic feedback, user experience as a priority and documentation.

7.1.2. Current state of collaboration during Open API development lifecycle

The current sate of collaboration during the Open API development lifecycle appeared as a topic in both literature review and as a research question. While for the literature review it was important to see where in processes collaboration between API providers and consumers is advised, in the interviews the goal was to have a look at the real state of collaboration in the industry at the moment.

Results of this question appeared as in a very interesting form - all of our interviewed experts from the API provider side have done the collaboration with their consumers, while none of our interviewed experts from the API consumer side ever had a chance to be involved in any kind of collaboration besides solving the concrete issues in integrating the API, which were solved through different forums or issue tickets. More detailed analysis which is listed below can provide some insights why is development done that way:

Provider side:

- **More efficiency for providers** - On the providers side, this reason was stated in almost all interviews done (I3, I4, I8, I9, I11) and basically means that for the providers it is a lot more efficient to work in a collaborative way with customers who they already know and have some past experience working with, especially during integrating the API into their business. For the consumers, interviewed experts stated that having a much larger number of consumers involved makes it very hard to manage, and that a lot of smaller consumers who use API on a smaller scale do not seem interested to invest a lot of time into participating in this kind of collaborations.
- **Smaller consumers still have a chance to participate** - Even though it is not on a large scale, besides the regular ticketing and troubleshooting ways of collaboration on solving development issues, interviewed experts from the provider side (I3, I4, I5, I8,

I9, I11) involved their consumers in different ways. It included survey on developer portal, about developing the new feature which was aimed to use time only of those consumers that are interested. It was achieved using a gateway question and which ended the survey immediately in case users were not interested in the feature. Other ways included involvement of consumers in hackatons, where the goal of API providers was to monitor the usage and efficiency in integrating their APIs, having developers which have non or very limited experience of using it in the past.

Consumer side:

- **Lack of visibly offered collaboration mechanisms** - After getting the negative answers on their past involvement except for troubleshooting, all API consumers interviewed (I1, I2, I5, I6, I7, I10) were also asked about the existence and visibility of collaboration options on developer portals of API providers. The general feeling of all interviewed API consumers was that collaboration options are not offered at all or they are not visible enough for them to participate.
- **Existence of alternatives** - From the perspective of consumers (I1,I2,I5,I6,I7), if their API is not business-critical which is usually the case, they are usually not interested in using their time to find a way to integrate the particular API, but they would rather move to an alternative. It also showed that API consumers are more interested in having the API which is already efficient and easy to use, rather than being involved in improvements.

Current state of collaboration shows that there is a lot of potential for the improvements. Interviewed experts from the provider side agreed on that, stating that the current level of user involvement is not on the level it should be (I3, I9). Also, the important information gathered during this interview is that developing APIs with more consumer involvement is currently in the focus of some companies, and that early adopters already start to involve consumers on a much higher level than in the past.

7.1.3. API Development lifecycle phases that should be collaborative

The following analysis category came from the idea to have a different view on the research results from the literature review and to analyze the possible similarities and / or differences between findings in terms of collaboration. Interviewees were presented with the research results in the form of identified phases, but without showing them which of these phases were found as collaborative. The goal here was to make the least possible influence on their answers and their opinions on the collaboration during API development lifecycle.

It is important to mention that the details of analysis varied between interviews, and that sometimes interviewees did not explain or even mention collaboration in phases which are collaborative in their nature, like marketing and engage (onboarding). The list below presents collaboration during API development lifecycles based on the interview results:

- **API Strategy** - This phase of API development lifecycle provided different opinions, a part of our interviewees thought that collaboration should exist in this phase (I1, I3, I7, I9, I11), while others were against it (I2, I5, I6). On the side of supporting collaboration in this phase, the main arguments included involving your business partners from consumers in market research to form the strategy and see if there is a need to build the API. Arguments against were around the statement that API strategy as a part of company's business strategy should stay internally, and it should not be done with any consumer involvement.

- **Design** - The only phase of API development lifecycle where the full consensus for collaboration was reached in all of the conducted interviews (I1, I2, I3, I4, I5, I6, I7, I8, I9, I10, I11). It was identified that not only this phase is important, but it should also be improved, as a large number of API providers still provides their API from inside out perspetive, without considering consumer needs in early stages. This kind of involvement was considered to be done in two different ways, one is to actually involve them in design process itself, other is to verify the created design through the mockup. Interviewees that identified the latter way also shared the opinion that design and mock / simulate phases should be merged as design itself is actually mocked, but since it was also found out that some of companies of our interviewed experts don't use the concept of mocking their API, it was still kept as a separate phase for collaboration.

- **Mock / Simulate** - The phase which brought a certain degree of uncertainty to our interviewees in terms of collaboration because, as explained in design phase, a lot of them couldn't differentiate and separate the two. In the end, the results showed that interviewees (I1, I3, I5, I10) would keep mock phase as a separate with the need to collaborate, while interivewees (I2, I6, I9, I11) saw this phase as a joined phase with design. No opinions against collaboration in this phase were collected.

- **Build, Secure, Publish** - These three phases are grouped together as all of the interviewed experts agreed that they should be done internally.

- **Test** - Different opinions on collaboration in test phase were also collected during the interviews. In interviews (I1, I2, I5, I9, I10) participants saw the potential for collaboration between API providers and consumers, where the most important reasons included the feedback before the API is published to larger group of consumers is key, in order to assure that the issues which may appear are spotted on time. The opposite opinion was collected in interviews (I3, I4, I8) and it mostly came as a direct consequence of user involvement in early phases such as design, meaning that the rest of work with testing can be done internally.

- **Documentation** - Also provided different opinions, while the majority of interviewees (I1, I2, I3, I6, I9, I10, I11) saw themselves collaborating in documentation writing, especially in later stages where they would like to provide their own contributions to documentation parts which need to be improved, there was still other opinion in interview (I5) where the conclusion was that good enough documentation should be provided entirely on the provider side, with possible feedback incorporated, but without systematic way of collecting it.

- **Onboard and Engage, Marketing the API** - These two phases provided the similar way of responses, as they are both directed to the consumers they are collaborative by nature. This kind of opinion appeared in a number of conducted interviews (I1, I2, I3, I9, I11), while in other interviews these phases were not mentioned or further analyzed. Important outcome of the analysis of these phases was also that consumers could be involved in evaluating these phases and provide potential improvement ideas.
- **Version** - Widely accepted as the phase where collaboration is needed by interviewees (I1, I2, I4, I6, I7, I8, I9, I10, I11), includes two ways of collaborating - the first one is the direct collaboration and involving users in idea collection and considering their needs for the new API features, the second one is coordination in case of changes which might be breaking and might affect consumers' integration.
- **Monitor and Analyze** - Another phase which was confirmed as the one that should be done internally.
- **Monetization** - In this phase the larger number of participants was against consumer involvement (I4, I2, I5, I6), sharing the opinion that it is a part of decisions which should be internally kept to the provider company, and that consumers should only choose from the models offered or search for the alternative solution. On the other hand, in interviews (I1, I3, I9) the 'voice of consumers' was seen as the important part of monetization strategy, as it can give API providers a better sense of how high they can go with prices, and what models are their consumer more likely to use.
- **Retirement plan** - Plan to retire the API in its entirety or one of its versions is seen by majority of interviewees (I1, I2, I3, I4, I5, I10, I11) as a business decision on the side of providers, which should only be communicated with interested sides from consumers. Consumers should be given a certain period to move to newer version or to other API. One of the main reasons for that is that retirement often comes in a period when the number of users reaches very low levels. Another opinion was provided by interviewee (I9), who thought that only informing is not enough and that there should be a certain group of users which should be involved in retirement plan, as it may not be easily visible that some users have important use cases based on the provided API.

7.1.4. API Provider team roles involved in the collaboration

The goal of this analysis category was to have an overview on the current situation regarding API provider roles which are involved in collaboration with consumers. Unlike the part with API development lifecycle phases, for API provider roles interviewees were not provided with the list of identified roles from the Chapter 6, as the goal was not to influence their answers and possibly identify the new roles which didn't appear in the literature, as for this part of literature review the number of offered sources was a lot smaller than for lifecycle phases.

Given that none of our interviewees had a chance to collaborate with API providers in the past, they were asked to think about potential roles that could be involved in the collaborative phases of API development. Similar to previous analysis categories, results are presented in the form of a list:

- **Developer** - In the interviews (I1, I2, I6, I7, I8, I10, I11) developers were primarily identified as roles which should collaborate with consumers on technical issues, but also as some of the interviewees saw them directly involved in collaboration, mostly as the role that has a lot of technical knowledge and someone who can easily understand with developers from the consumer side . Interviewee (I4) saw the collaboration only on the technical level and if API is not enough large enough in its consumption, otherwise there should be a separate support person dedicated to doing it. Finally interviewees (I3, I5, I9) saw no developer involvement in direct collaboration - meaning they should concentrate on internal topics while others can cover collaboration.
- **API Product Manager**- Identified from the interviewees (I1, I2, I3, I4, I5, I6, I7, I8, I9, I11) this role that combines technical and management skills is seen as the most important collaborative role. This should be the main role to work with consumers when it comes to new functionalities, and to strategical decisions about the existing functionalities and endpoints.
- **API Evangelist** -API Evangelist or a special role dedicated to working with external developers community did not appear often as needed during the interviews. Two of the interviewees (I3, I11) saw this role as important but only once the API is mature enough, while interviewee(I9) saw this special role needed from the first day, seeing the potential to relax the workload of developers and architect. Finally, interviewee (I2) provided the opinion that this role is not needed, as they are often decoupled from the actual product and it brings another level of communication in between that makes relations between providers and consumers more complicated.
- **API Architect** Identified as collaborative in interviews (I3, I5, I8, I11) this is a role that is often covered by other team members like a developer, and it covers collaboration mainly when it comes to the design, both initially and in versioning processes. On the other hand interviewee (I9) saw architects together with developers behind the actual collaboration work, having more focused on internal topics during designing and building of the API.
- **Support and operation** Role in the API team which should communicate and react on consumer tickets was seen as important in two interviews (I3, I7, I10), while one interviewee (I4) saw this role appearing and being collaborative by its nature only once API is large enough that developers can not handle.
- **Legal** - Identified as the collaborative role only once (I7), this role should collaborate with users who have different legal requirements and law systems in their countries, like in the case of GDPR.

8. Discussion

In the discussion chapter differences between findings of collaboration in literature and interviews are presented. It also contains ideas for improvements for the collaborative development of Open APIs.

8.1. Similarities and differences in findings - Collaboration

The analysis for collaboration in different API development lifecycle process phases was performed independently in literature review and interviews analysis stages of this research, based on the same input - identified phases during the literature review. When two stages were read side by side, it turned out that collaboration relations in identified phases can be grouped into four different types of relations:

- **Collaboration does not exist in both literature review and interviews analysis: Build, Secure, Publish, Monitor and Analyze** - These phases were identified as ones that should entirely be done on the provider side. Build and Secure phases because previous phases were already collaborative and all conclusions needed to be made involving consumers were already made. Publishing as a step to make the API available to all users was also identified as a business decision that should not involve consumers. Finally, Monitor and Analyze depends on metrics and key performance indicators that are created without consumer involvement.
- **Collaboration exists in both literature review and interviews analysis, in the same or similar way: Onboard and Engage, Marketing the API, API Strategy, Version, Monetize** - In all of these phases, literature review findings and methods of collaboration were confirmed during interviews. Onboard and Engage, as well as Marketing the API were seen as phases which are self explanatory, as both of them are directed to consumers and their involvement needs to exist. Version phase was seen as one of the most important for collaboration, especially once the API is published. In this phase, all of the changes, issues and improvements are discussed and the user involvement is of large importance as all the breaking changes and possible issues with user experience should be avoided. Lastly, Monetization phase was identified in the literature as collaborative, and the interviewees who identified it as collaborative mostly identified the same principals for how this collaboration should happen.

- **Collaboration exists in both literature review and interviews analysis, in different ways: Design, Mock / Simulate, Retirement Plan** - In design phase the first significant difference was spotted between literature reviews and interviews, and that difference is based on interviews where it was stated that consumers should be directly involved in design process through active participation. On the other hand, literature review findings match with other group of interviews where the collaboration was seen as the process of design verification and ideas for design improvements in the form of mocked API functionalities. For the retirement plan, almost all interviewees who identified this phase as collaborative confirmed findings from the literature review that the decision on retiring the version or complete API should only be communicated to consumers. Still, there was a different opinion of one interviewee who saw consumers being actively involved in creating the retirement plan for one API or its version.

- **Collaboration exists in interviews analysis, but not in literature review: Testing, Documentation** - Here no information on the need for collaboration between providers and consumers was found in the literature, but in the interviews the situation was different, having many interviewees who stated the need for consumer involvement in them. For testing, collaboration was seen as essential in terms of providing final external feedback before public access to the API is allowed. Improving documentation and making its writing an outside-in process where improvements and changes should be user-driven, using direct collaboration in documentation pages of the developer portal.

In the end, seeing the major differences in only 2 out of 14 API development lifecycle phases, it becomes visible that most of the theoretical conclusions already exist in the business. Still, having these phases that do not have collaboration advised in the literature but the potential is seen by experts in this area, a number of processes that do not even include a lot of management related phases, and a lot of different opinions on whether some phases should be collaborative or not in both literature and conducted interviews shows that there is a large improvement potential in this area. Some of these ideas are presented in the next section of this thesis. It also confirms an opinion from the interview (I9) that the active collaboration between API providers and consumers only passed a stage of early adopters, and that the majority of the industry still needs to implement the best practices.

8.2. Ideas for improvements

In this section the final research question *RQ4: How could collaboration during Open API development be improved?* is answered. The answer to this research question is provided through the series of different improvement ideas that came out of previously conducted research and answering the previous research questions. Improvement ideas vary in their detail levels: From high level ideas like Improvement idea 6, to the concrete advises how the process of developing Open APIs could be improved like Improvement idea 1.

Improvement idea 1: Get to know your consumer better – conduct market research to identify groups of consumers that can be interested in the API and that can help identify the requirements better.

Challenge: This challenge was a direct outcome of the RQ1 where most common challenges during Open API development were identified. It was already shown that the most common challenge was that Open API consumers and their requirements remain unknown to API providers. In the end, this makes requirements identification and the whole design process a lot harder, as API providers need to think themselves about use cases, user personas and do the best guess, which in the end can result in the API which was not built according to user needs.

Proposed improvement: The solution for this issue should be in performing detailed market research, given the initial concept of the API and the field of business where the planned consumers should be. That way, a clear picture of potential API consumers can be viewed and therefore a better choice of consumers can be made out of the existing business partners. If partners in identified fields do not exist, this approach also allows providers to spend less time searching for a suitable consumer for collaboration and to get in communication with the ones which are more likely to be interested in using the API.

Improvement idea 2: Think about monetization as early as in the API Strategy phase – even if API is not planned to be monetized immediately.

Challenge: The literature review results showed that almost half of the identified API development processes did not even consider monetization as a separate phase. Also, in processes where monetization was considered, it was not clear when this phase should happen - early in the API development process or only after the API has been available to consumers for a certain period and usage parameters showed that there is a large number of requests and integrations of the API.

Proposed improvement: To get a clearer idea about this challenge, our interviewees were also asked about the potentials and stage where monetization should be included, which in the end resulted in this improvement idea. Interview results highly influenced the shaping of this improvement idea. Open APIs should always be considered as a possible source of income, and API providers should consider possible strategies for monetization even before the design process starts. Monetization strategy should be created in parallel or immediately after finalizing the API strategy. Even if there are no immediate plans for the API to be monetized, a clear strategy should exist, as it is essential for product-oriented thinking about the API. This approach also allows a simple switch to monetization options once the API is ready to be monetized.

Improvement idea 3: More is less – spend more time on strategy and design early in the lifecycle than on change management later in the lifecycle

Challenge: Another challenge that came out of RQ1 and the most common challenges during Open API development. Interviewees identified change management as one of the hardest topics during API lifecycle after publishing, meaning that knowing that API is being used by a large group of consumers. It is dangerous to create any changes which may break their integrations or that will require the API to be down for a certain period, creating a large number of unhappy consumers and bad reputation of the API.

Proposed improvement: It is essential to create the API that is tailored to consumer needs, as it needs less larger-scale changes during the API lifecycle after publishing. This is achieved through spending more time in the design phase when possible errors and bugs in API functionality are easier to resolve. It also includes creating mocks that should be given out to identified potential consumers and building their feedback into the design and later the API product itself.

Improvement idea 4: Encourage smaller-scale consumers to take part in collaboration – do not collaborate only with larger-scale partners.

Challenge: The literature review showed to a smaller extent and interviews showed to a larger extent - most of the API providers have a smaller group of larger-scale partners that are involved during collaboration, especially in essential earlier phases like API strategy and design. In the end, the majority of smaller-scale consumers such as startups do not even have a chance to give their opinion except for troubleshooting and concrete bugs during the API integration process.

Proposed improvement: Smaller-scale consumers should be included a lot more in collaboration. This can be achieved using different parts of the developer portal where improvement ideas could be collected, and directly involving the proposer of an interesting idea in the design and development process. Also, the developer portal should include visible pop-up windows where surveys on new features that are about to be implemented could be regularly presented. The same should be with the user satisfaction survey on newly published features or endpoints. Finally, there is an idea of competitions such as hackathons where consumers with less or no experience using the API that can explore it and API providers can have a direct look at how their API is being used. This idea was presented in some of the interviews and should be used more often.

Improvement idea 5: Outside-in documentation involving consumers – make your documentation interactive, implement, and reward the consumer efforts in documentation improvement.

Challenge: During the literature review, documentation was in the most of identified processes considered as a part of another phase, mostly development. Also, it was not considered as a phase where collaboration should happen. The results of interviews on the other hand showed a lot of potential for collaboration in that are, and criticized the current state of Open

API documentation where they are usually just presented to consumers without any potential options for collaboration and user contribution. Even though some API documentation are moving in the right direction regarding this topic, it still remains a large challenge for most of the players in the market.

Proposed improvement: Already mentioned in the challenge description, API providers that do not use interaction in their documentation should start. It includes questioning consumers about their satisfaction with the presented documentation and potential contribution to improving the documentation. Consumers whose improvement ideas were accepted should be informed that their contribution was accepted, and if possible they should also be rewarded. There is a wide range of possible rewards, from material rewards such as in the form of promo material or similar to a simple recognition in that documentation page stating who helped improve it.

Improvement idea 6: Focus on management topics together with technical – constantly improve your API development process.

Challenge: Identified API development processes showed another interesting insight - all of the identified processes include technical phases of the API lifecycle such as build, secure or monitor and analyze. Where the difference mostly stands are phases connected with management such as marketing the API, API strategy or monetization.

Proposed improvement: The above-identified challenge shows potential for improvements in this area too, but since it is a wide range of topics that can be included, it becomes challenging to come up with concrete improvement ideas. It is only possible to advise API providers to focus on productizing their API by making its development process closer to the regular product development process and constantly improving it according to the current trends.

Improvement idea 7: Have clear roles dedicated to collaboration with consumers – create a stable group of team members that combine technical and business roles to collaborate with consumers.

Challenge: In the final question of interviews our interviewees were asked to identify the potential roles from the API provider side to be involved in the collaboration with consumers. In the end, the mixed opinions which were very often clashing created, most often about collaboration in early phases of API development lifecycle such as design, where it was not easy to get a uniform opinion. It is important to mention that for issues during the API usage there was no such dilemma and that interviewees either pointed developers in case the API is smaller in the number of users, or special team members for support if the API is larger.

Proposed improvement: Even though it impossible to propose a perfect group on the side API provider for the collaboration with consumers, because different teams have different roles and responsibilities, the improvement idea here is oriented on having a group with combined technical knowledge and business knowledge, that would for example include technical roles like API architect or API developer, but also roles connected with management, API Product Manager in earlier stages, and a role like API Evangelist or Community Manager in later stages when API grows in the number of users. This kind of organization for collaboration on the API provider side will allow that no technical details are missed, but also that requirements are understood on business side.

9. Conclusion

The final section of this thesis contains the summary of research results that were found during this thesis. Limitations that were faced during the work on this thesis are also presented, as well as potential ideas for the future work related to this thesis.

9.1. Summary

In the summary section four Research Questions (RQ) will be presented again, together with summarized answers to them.

RQ1: What are current challenges during Open API development?

This research question was answered through semi-structured interviews and its most important findings are presented in Chapter 7. The importance of this research question was to validate already identified challenges that were used as the motivation for this work. Our interviewees identified 9 different challenges during the Open API development, with more than half of them related to the lack of collaboration. Therefore, the initial assumption that a lack of collaboration between API providers and consumers can be seen as the major issue during Open API development processes was confirmed.

RQ2: What are the phases that the development processes of Open APIs should contain?

The second research question was addressed through an extensive literature review and parts of it were further addressed during interviews. In the end two important results came out of it, presented in Chapter 5. First is the concept matrix that analyzed different API development processes and the phases they contain. In the end, from 13 identified processes 14 different phases and their descriptions were identified. Additionally, each of these phases was analyzed for the existence of collaboration, and the question to identify collaborative phases was also asked during interviews, with results presented in the Chapter 7. Finally, these outcomes were compared for similarities and differences in the Chapter 8.

RQ3: What stakeholders are involved in the collaborative development of Open APIs?

Answering this research question began with stakeholders analysis was a part of API value chain analysis in the Section 2.2.3. Further literature review showed that more detailed analysis was required on the API provider side of stakeholders. This was done in Chapter 6, where in the similar way to processes 13 different roles were identified and matched with 7 sources that provided them. It was also important to analyze which of the roles from API provider side could be collaborative. It was achieved during interviews, where a specific question was asked to review the API provider roles that were involved in provider-consumer collaboration in the past, or if which of them could be involved in the future in case the interviewee had no experience with collaboration. These results were presented in the Chapter 7 of this thesis.

RQ4: How could collaboration during Open API development be improved?

The final research question was addressed summarizing the results from previous three research questions. This approach helped to come up with improvement ideas from different perspectives, from the concrete ideas that came out of challenges from RQ1, to some more higher-level recommendations that could be identified from RQ2 and RQ3. In the end, 7 different improvement ideas were presented, having the challenge explained first and then the proposed improvement. Improvement ideas were presented in the Chapter 8 of this thesis.

9.2. Limitations

During the work on this thesis, a few major limitations influenced the research and its results.

Firstly, the lack of reflated scientific work and structured API processes created a lot of processes needed to be standardized in their terminology and content of their phases, which impacted analysis quality.

Secondly, due to time constraints only a limited number of interviews could be carried out, which affected the validity of their results which could be improved in case of a larger number of interviews and having more analysis material. Also, one more limitation connected to time constraints was that ideas for improvements could not be validated.

Finally, with the global situation related to coronavirus pandemic, the final two months of this research were also largely affected. The most important issues were the lack of the option to do interviews with experts face to face, where the goal and questions could be explained differently and collecting the information would be a lot easier, and the lack of face to face meetings with the advisor, which in the period before the pandemic were very efficient and with lots of impact on the work.

9.3. Future work

The time limitations that were presented in the previous leave a space for different future work that could validate and improve findings from this work.

Improvement ideas need to be validated through a separate study where experts in the API filed could give their opinion on the potential for their implementation and also potential ways to make them more effective. Besides that, a concrete implementation and measuring success could also be useful for validating these ideas.

Identified API development process phases and API provider roles can also be used for further research. Previously mentioned as one of the limitations, phases, and roles identified during this work can now be challenged in the future research through the validation process and further development of their definitions by working with concrete examples in organizations.

Finally, since challenges in the API domain change dramatically every few years, as it could be seen in the research from (Smartbear 2019), it is important to track the available data and work on the ideas for improvements for new challenges that may appear in coming years. This kind of future work can also show the development of collaboration between API providers and consumers as an identified challenge.

A. Interview Questionnaire and Guide

Interview Questions - Collaborative Development of Open APIs – Status Quo and Ideas for Improvements – Master Thesis by Mihailo Rajacic

Introduction:

In the world of API Economy we can find that an average of 2019 new APIs is added every year (Santos 2019) and the growing trend has been happening continuously for over a decade. Besides the positive changes happening together with the growth, there are also certain ''challenges enterprises face in capitalizing on the potentials of API ecosystems" (Vukovic et al. 2016). Some of these challenges include:

- Lack of API related knowledge in companies that are not digital companies (e.g. Google, Amazon, Yahoo)
- Open API design and development currently doesn't include enough involvement from consumers.

To address these challenges, our study aims to identify the different phases of an open API Lifecycle process and whether these phases are or should be collaborative. Also, we aim at identifying the key stakeholders in open API development, as well as roles and responsibilities in API teams. In this thesis focus on open APIs which are:

- Web – APIs which are accessed over the web using the HTTP protocol. (RapidAPI 2019)
- Open - APIs that are accessible to a broad range of developers. We do not consider APIs used internally or only by a small group of selected partners (Moilanen 2017)

Interview Purpose

In this interview the goal is to discuss current challenges in API development and collaborative aspects of API development.

Terms of Confidentiality & Tape Recording

This interview is conducted anonymously. Only a classification of your company and your role, which we align with you, will be related to the results. We would like to record this interview so that we can derive the greatest value from your expertise within the framework of our research work.

Do you agree to recording of this interview? YES / NO (including deletion of the audio recording after transcription).

Format of the Interview

This interview consists of five open questions. The interview is planned to last 30-45 minutes, depending on the details of answers and the direction in which the conversation will go.

Contact & Last Remarks

If you have any further questions or comments following this interview, please feel free to contact Gloria Bondel (gloria.bondel@tum.de) or Mihailo Rajacic (mihailo.rajacic@tum.de).

Do you have any other questions before the interview starts?

Company / Interviewee info:

1. Which industry does your company belong to?
2. How much experience does your company have with open APIs (number of APIs provided / consumed)?
3. What is your role in your current team?
4. For how many years have you been working in the field of API Development / API Management?

Research related questions

1a. (consumers) What are your experiences with using open APIs? (How often do you use them)
1b. (providers) What are your experiences with providing open APIs? (How often do you use them)
2. What challenges do you see in Public API development compared to Private API development?
3a. (consumers) In the past, how did you collaborate or communicate with providers of open APIs?
3b. (providers) In the past, how did you collaborate or communicate with consumers of open APIs?
4. Seeing the identified API development process phases (from the research), where do you see possible options for collaboration?
5a. Which roles did you collaborate with (from API Providers)? In case you did not collaborate in the past, which roles do you see suitable for collaboration?
5b. (providers, after the first interview with providers changed for 5a) Seeing the identified roles and responsibilities (from research), which roles do you see in your API team? Which roles do you find unnecessary?

List of Figures

List of Tables

Bibliography

Anthony, A. (2016). *Tracking the Growth of the API Economy*. https://nordicapis.com/tracking-the-growth-of-the-api-economy/. Accessed: 2020-05-10.

Santos, W. (2019). *APIs show Faster Growth Rate in 2019 than Previous Years*. https://www.programmableweb.com/news/apis-show-faster-growth-rate-2019-previous-years/research/2019/07/17. Accessed: 2019-12-04.

Iyer, B. and M. Subramaniam (2015). *The Strategic Value of APIs*. https://hbr.org/2015/01/the-strategic-value-of-apis. Accessed: 2020-04-04.

Evans, P. and R. Basole (2016). "Revealing the API Ecosystem and Enterprise Strategy via Visual Analytics". In: *Communications of the ACM, February 2016, Vol. 59, No. 2, pp 26-28*.

Panetta, K. (2016). *10 Management Techniques from Born-Digital Companies*. https://www.gartner.com/smarterwithgartner/10-management-techniques-from-born-digital-companies/. Accessed: 2020-05-10.

Pamnani, H. (2017). *'Become Digital' if Not 'Born Digital'*. https://www.entrepreneur.com/article/298535. Accessed: 2020-05-10.

Vukovic, M. et al. (2016). "Riding and Thriving on the API Hype Cycle". In: *Communications of the ACM, March 2016, Vol. 59, No. 3, pp 35-37*.

Smartbear (2019). *The State of API 2019*. https://static1.smartbear.co/smartbearbrand/media/pdf/smartbear_state_of_api_2019.pdf. Accessed: 2020-05-10.

Bondel, G. et al. (2019). "Towards a Process and Tool Support for Collaborative API Proposal Management". In: *Twenty-fifth Americas Conference on Information Systems, Cancun, 2019*.

Smith, T. (2018). *DZone Research: API Management Issues*. https://dzone.com/articles/dzone-research-api-management-issues. Accessed: 2019-12-04.

Glickenhouse, A. and L. England (2016). *API Monetization – Understanding your Business Model Options*. https://www.ibm.com/downloads/cas/L5Q82XR0. Accessed: 2020-02-10.

Womack, K. (2016). *Building an API Team*. https://blog.hitchhq.com/building-an-api-team-4b0aa33e856e. Accessed: 2020-02-23.

Bell, A., S. Rensen, L. Weir, and P. Wilkins (2018). *Implementing Oracle API Platform Cloud Service*. Packt Publishing.

De, B. (2017). *API Management – An Architect's Guide to Developing and Managing APIs for Your Organization*. APress.

Hussain, F., R. Hussain, B. Noye, and S. Sharieh (2019). "Enterprise API Security and GDPR Compliance: Design and Implementation Perspective". In:

RapidAPI (2019). *How To Use an API (The Complete Guide)*. https://rapidapi.com/blog/how-to-use-an-api/. Accessed: 2019-11-30.

Zhu, W.-D. et al. (2014). *Exposing and Managing Enterprise Services with IBM API Management*. IBM Redbooks.

Moilanen, J. (2017). *From private to public — API types from business perspective*. `https://medium.com/apinf/from-private-to-public-api-types-from-businessperspective-76b6e6e12624`. Accessed: 2019-12-09.

Ashby, D. and C. T. Jensen (2018). *APIs for Dummies, 3rd IBM Limited Edition*. John Willey & Sons.

Siriwardena, P. (2014). *Advanced API Security: Securing APIs with OAuth 2.0, OpenID Connect, JWS, and JWE*. APress.

Patni, S. (2017). *Pro RESTful APIs*. APress.

Doerrfeld, B. C., B. Pedro, K. Sandoval, and A. Krohn (2015). *The API Lifecycle, An Agile Process for Managing the Life of an API*. Nordic APIs.

Niinioja, M. (2018). *What is an API? A product, service, integration or something else?* `https://www.osaango.com/blog/what-is-an-api-a-product-service-integration-or-something-else`. Accessed: 2019-12-15.

Myllärniemi, V., M. Raatikainen, and M. Komssi (2014). "The Role of Platform Boundary Resources in Software Ecosystems: A Case Study". In: *2014 IEEE/IFIP Conference on Software Architecture, pp 11-20*.

Endler, M. (2017). *How APIs Become API Products*. `https://medium.com/apis-and-digital-transformation/how-apis-become-api-products-580b5c01b96f`. Accessed: 2019-11-20.

Holley, K. et al. (2014). *The Power of the API Economy Stimulate Innovation, Increase Productivity, Develop New Channels, and Reach New Markets*. IBM.

Google (2018). *The API Product Mindset*. Google.

CA-Technologies (2015). *API Strategy and Architecture: A Coordinated Approach*. `https://docs.broadcom.com/doc/api-strategy-and-architecture-a-coordinated-approach`. Accessed: 2020-02-10.

Jacobson, D., G. Brail, and D. Woods (2012). *APIs: A Strategy Guide*. O'Reilly Media, Inc.

Queensland, U. of (2019). *Digital Transformation*. `https://digital-transformation.uq.edu.au/home/central-data`. Accessed: 2020-04-10.

Tan, W., Y. Fan, A. Ghoneim, M. A. Hossain, and S. Dustdar (2016). "From the Service-Oriented Architecture to the Web API Economy". In: *IEEE Internet Computing, Volume: 20 , Issue: 4 , July-Aug. 2016, pp 64-68*.

Bush, T. (2019). *What Is The Difference Between Web Services and APIs?* `https://nordicapis.com/what-is-the-difference-between-web-services-and-apis/`. Accessed: 2019-12-15.

Suhardi, S., Y. Purnomo, and R. Doss (2015). "Service Engineering Based on Service Oriented Architecture Methodology". In: *TELKOMNIKA, Vol. 13, No. 4, December 2015, pp. 1466 1477*.

Greca, P. L. (2014). *APIs: A new path to SOA*. `https://blogs.mulesoft.com/dev/api-dev/soa-and-api/`. Accessed: 2019-12-15.

Mitchell, E. (2018). *APIs versus services – is there a difference?* `https://developer.ibm.com/technologies/api/articles/api-vs-services-whats-the-difference/`. Accessed: 2019-12-19.

Refalo, O. (2016). *API, Managed API vs SOA*. `https://www.linkedin.com/pulse/api-managed-vs-soa-olivier-refalo/`. Accessed: 2019-12-15.

Cardoso, J., K. Voigt, and M. Winkler (2008). "Service Engineering for the Internet of Services". In: *Enterprise Information Systems, 10th International Conference, ICEIS 2008, Barcelona, Spain, June 12-16, 2008, Revised Selected Papers, pp 15-27*.

IJCE (2018). *International Journal of Collaborative Engineering - Introduction*. `https://www.inderscience.com/jhome.php?jcode=ijce`. Accessed: 2020-04-19.

Johnson, W. H. and R. Filippini (2009). "Internal vs. External Collaboration: What Works". In: *Research Technology Management, Volume 52, Issue 3, 2009, pp 15-17*.

Ellis, C. A., S. J. Gibbs, and G. Rein (1991). "Groupware: some issues and experiences". In: *Communications of the ACM, January 1991, Vol 34, No 1, pp 38-58*.

Borghoff, U. M. and J. H. Schlichter (2000). *Computer-Supported Cooperative Work*. Springer.

Leimeister, J. M. (2014). *Collaboration Engineering*. Springer.

Fremantle, P., J. Kopecký, and B. Aziz (2016). "Web API Management Meets the Internet of Things". In: *Lecture Notes in Computer Science book series (LNCS), Volume 9341, pp 367 - 375*.

Uddin, G. and M. P. Robillard (2015). "How API Documentation Fails". In: *IEEE Software, Published by the IEEE Computer Society, July/August 2015, pp 68-75*.

Costa, J. (2018). *Improve API Performance with Caching*. `https://developer.akamai.com/blog/2018/05/31/improve-api-performance-caching`. Accessed: 2020-03-20.

Myers, B. A. and J. Stylos (2016). "Improving API Usability". In: *COMMUNICATIONS OF THE ACM, June 2016, Vol. 59, No. 6, pp 62-69*.

Doerrfeld, B. et al. (2016). *How to Successfully Market an API*. Nordic APIs.

Akana (2015). *Building Successful APIs*. Akana.

Beverungen, D., H. Luttenberg, and V. Wolf (2018). "Recombinant Service Systems Engineering". In: *Bus Inf Syst Eng, 2018, Vol. 60, No.5, pp 377–391*.

Watson, R. T. and J. Webster (2002). "Analyzing the Past to Pretare for the Future: Writing a Literature Review". In: *S Quarterly, Vol. 26, No. 2 , pp 13-23*.

Massé, N. (2019). *Full API lifecycle management: A primer*. `https://developers.redhat.com/blog/2019/02/25/full-api-lifecycle-management-a-primer`. Accessed: 2019-12-15.

Broadcom (2017). *API Strategy and Design - Your First Stop in Full Lifecycle API Management - Layer 7® API Management*. `https://www.broadcom.com/sw-tech-blogs/api/api-strategy-and-design-your-first-stop-in-full-lifecycle-api-management-layer-7-api-management`. Accessed: 2019-12-20.

Vester, J. (2017). *RESTful API Lifecycle Management*. `https://www.axway.com/en/products/api-management/full-lifecycle-api-management`. Accessed: 2020-01-15.

Adams, W. (2015). "Conducting Semi-Structured Interviews". In: *Handbook of Practical Program Evaluation, Edition: 4, Chapter: Conducting Semi-Structured Interviews, pp 492-505*.

Schmidt, C. (2004). "The Analysis of Semi-structured Interviewss". In: *A Companion to QUALITATIVE RESEARCH, 5D: ANALYSIS, INTERPRETATION AND PRESENTATION, Section 5.10, pp 253-258*.

Axway (2019). *Full Lifecycle API Management*. `https://www.axway.com/en/products/api-management/full-lifecycle-api-management`. Accessed: 2020-01-10.

Arnny, Y. (2017). *MuleSoft API Lifecycle Management*. `https://dzone.com/articles/mulesoft-api-management-life-cycle`. Accessed: 2019-12-18.

TIBCO (2019). *Success Guide for a Healthy API Program.* `https://apifortress.com/wp-content/uploads/2019/08/API-Fortress-TIBCO-Mashery-Guide-for-a-Healthy-API-Program-eBook.pdf`. Accessed: 2019-12-18.

TransparentData (2019). *API Lifecycle. Batteries not included cause no needed.* `https://medium.com/transparent-data-eng/api-lifecycle-batteries-not-included-cause-no-needed-efb54ab86722`. Accessed: 2020-01-14.

Bortenschlager, M. and S. Willmott (2017). *The API Owner's manual*. 3Scale.

Preibisch, S. (2018). *API Development: A Practical Guide for Business Implementation Success*. Nordic APIs.