API-m-FAMM: a Focus Area Maturity Model for API Management*

Michiel Overeem[a,b,*], Max Mathijssen[a] and Slinger Jansen[b,c]

[a]AFAS Software, Inspiratielaan 1, Leusden, The Netherlands
[b]Utrecht University, Princetonplein 5, Utrecht, The Netherlands
[c]Visiting Scientist, School of Engineering Science, LUT University, Finland

ARTICLE INFO

Keywords:
API Management, Maturity model, Focus Area maturity models

Abstract

Context: Organizations are increasingly connecting software applications using Application Programming Interfaces (APIs) to share data, services, functionality, and even complete business processes. However, the creation and management of APIs is non-trivial. Aspects such as traffic management, community engagement, documentation, and version management are often rushed afterthoughts.
Objective: In this research, we present and evaluate a focus area maturity model for API Management (API-m-FAMM). A focus area maturity model can be used to establish the maturity level of an organization in a specific functional domain described through a number of areas. The API-m-FAMM addresses the areas Lifecycle Management, Security, Performance, Observability, Community, and Commercial.
Method: The model is constructed using established methods for the design of a focus area maturity model. It is grounded in literature and practice, and was developed and evaluated through a systematic literature Review, eleven expert interviews, and five case studies at software producing organizations.
Result: The model is described in detail, and its application is illustrated by six case studies.
Conclusions: The evaluations are reported on, and show that the API-m-FAMM is an efficient tool for aiding organizations in gaining a better understanding of their current implementation of API management practices, and provides them with guidance towards higher levels of maturity. The detailed description of the construction of the API-m-FAMM gives researchers an example to further support the available methodologies, specifically how to combine design science research with these methodologies. Additionally, this study's unique case study design shows that maturity models can be successfully deployed in practice with minimal involvement of researchers. The focus area maturity model for API Management is maintained on www.maturitymodels.org, allowing practitioners to benefit from its useful insights.

1. Introduction

In recent years, there has been an increasing demand among organizations to have access to enterprise data through a multitude of digital devices and channels. This demand is also increased by the transformation from software product towards a platform, called 'platformisation' Poell et al. (2019). Platforms are a vehicle for software ecosystems and are defined as a set of organizations collaboratively serving a market for software and services (Jansen et al., 2013). In order to meet these demands, enterprises need to open up and provide access to their assets in an agile, flexible, secure and scalable manner (De, 2017). These assets include matters such as raw and cleansed data or functionality that performs complex calculations or data processing based on inputs (Weir, 2019). Access to these assets may be provided by utilizing Application Programming Interfaces (APIs). De (2017) defines an API as a software-to-software interface that defines a contract for applications to communicate with one another over a network, without the need for any user interaction.

As shown by an analysis conducted by ProgrammableWeb (Santos, 2019), known as the largest directory of APIs, the usage and offering of APIs has evolved from a curiosity to a trend since 2005. This observation is further supported by a survey conducted by Coleman Parkes Research (2017), showing that 88% of global enterprises have some form of an API program. Furthermore, the survey found that respondents experience a wide variety of benefits from their API programs, including an average increase in speed-to-market of around 18%. These statistics signal the emergence of the API Economy, in which organizations are offering access and the ability to recombine their digital services and products for novel value creation (Basole, 2019). As a result, by making their APIs accessible to external or partner consumers, these organizations are able to reach new markets, enable their business strategy and drive the creation of new innovative solutions (Bui, 2018). After an API has been created it needs to be managed so that developers may easily integrate it into their applications. API management is done by performing activities such as providing helpful documentation, controlling access to the API, as well as monitoring and analysing its usage.

Medjaoui et al. (2018) list three reasons that make it hard for organizations to improve their API management activities. Firstly, organizations that are performing well in terms of their API management programs often do not have the time, resources or personnel to share their experience and expertise with third parties. Secondly, organizations that

*This research was supported by the NWO AMUSE project (628.006.001): a collaboration between Vrije Universiteit Amsterdam, Utrecht University, and AFAS Software in the Netherlands.
*Corresponding author
michiel.overeem@afas.nl (M. Overeem); max.mathijssen@afas.nl (M. Mathijssen); slinger.jansen@uu.nl (S. Jansen)
ORCID(s): 0000-0003-4807-4124 (M. Overeem); 0000-0003-3752-2868 (S. Jansen)

are careful with regards to the amount of knowledge they share on their API management expertise might consider their know-how to be a competitive advantage, and will as such not feel urged to make their findings public. Finally, even in the event where organizations share their experience at public conferences, articles or blog posts, the information shared is usually company-specific and difficult to translate to a wider range of organizations' API programs.

Focus area maturity models (FAMMs) (Van Steenbergen et al., 2010; van Steenbergen et al., 2013) are an established method to communicate extensive domain knowledge. Not only do they contain this knowledge, they also offer a clear path for organizations to improve their maturity in a certain domain. In this article we present the API-m-FAMM, a focus area maturity model for API management. We show that this model improves on existing API management assessment frameworks and tools in terms of transparency and availability, and that it can be used by organizations that expose their API(s) to third-party developers to assess and evaluate their degree of maturity with regards to API management. We also extensively describe the methods that are applied in constructing the FAMM, and improve the available design methods by providing a concrete and detailed example. Additionally, explicit attention is paid to support organizations in performing their own assessment by using a *do-it-yourself* kit that we created and supplied to organizations.

Section 2 provides an overview of existing assessment models for API management and discusses their strong and weak points. A description of our research approach is given in Section 3, including the detailed steps that were taken to develop the FAMM. Section 4 describes the API-m-FAMM and Section 5 discusses how it is applied in four different companies. The results of these case studies are discussed in Section 6, while Section 7 discusses focus area maturity models in general. The treats to validity are discussed in Section 8. Section 9 summarizes our findings and contributions, among which are the previously undefined framework for API management, a detailed description of the construction of a focus area maturity model using both an existing methodology as well as tools from design science research, and finally an example of how we can make focus area maturity models more accessible by investing in their usability.

2. Related Work

In an effort to guide organizations in successfully managing their API programs, a number of commercial frameworks and tools exist with which organizations may evaluate and assess their API management approach and capabilities. In this section these existing frameworks, tools, models, reports and case studies are summarized and discussed. The existing frameworks and tools are discussed on several attributes. First of all we discuss availability, some frameworks are only available commercially and require extra costs. Secondly we discuss the grounding of the framework, some frameworks are grounded in scientific literature, others only in experience from an industry setting, and one framework is grounded in both. Finally we discuss the transparency of the framework: can we find details on how this framework is constructed. As becomes apparent, these frameworks are either not publicly available, transparent or grounded in academic literature.

Accenture API Management Suite - Based on their experience with implementing API programs, Accenture Technology Labs has developed an API Maturity Model (Tung, 2014). The Accenture model consists of 5 maturity levels, and is aimed towards helping organizations identify the maturity of their API management capabilities. These maturity levels are mapped onto five distinct dimensions, which details the processes an organization should implement in their journey from API enablement to industrialization. However, this maturity model fails to address certain core API management-related processes and capabilities such as versioning, threat protection and lifecycle management. The model is also quite out-dated, and has since been deprecated, as it is no longer available on Accenture's official website. Additionally, in part due to its industrial foundation and commercial nature, it is unclear as to how the contents of this model have been populated.

Endjin Maturity Matrix - This maturity matrix (Endjin, 2017) is a tool which was developed to aid business decision makers in assessing their organization's ability to evolve towards an API driven business model. The assessment is performed by having the organization fill out their perceived degree of maturity related to a set of categories, based on which practical suggestions for improvement are then provided.

While this maturity matrix comprises a selection of categories that are relevant in the scope of API management such as governance, documentation and support, it mainly focuses on strategies and commercial aspects. As a result, many API management-related aspects such as traffic management and community engagement are missing from the matrix.

WSO2 Platform Evaluation - In 2015, WSO2 published a whitepaper that describes digital business goals, outlines API oriented IT initiatives, and presents API management platform requirement categories (WSO2, 2015). Alongside this whitepaper, an evaluation matrix spreadsheet is provided that details a set of evaluation criteria, which may be used to evaluate API management platform vendors (Haddad, 2015). In illustrating the discipline of API management, two types of APIs are discerned: naked and managed APIs. A naked API is considered to be not monitored, managed, secured, documented or accessible through a self-service subscription portal, a managed API on the other hand is thought to be actively advertised and subscribe-able, available alongside a published service-level agreement (SLA), secured, authenticated, authorized, protected, as well as being monitored and monetized by using analytics. The whitepaper argues that to move from naked to managed APIs, the *API façade pattern* should be implemented, which enables teams to layer network-addressable endpoints, monitor usage, enforce usage limits, manage traffic, and authorize consumers. According to WSO2, an API management infrastructure should guide

teams towards best practices with regards to six main focus areas. However, it is unclear as to whether organizations are supposed to assign themselves scores, or whether they are assisted by WSO2 in this process. Furthermore, while a 'weighted score' column is included in the matrix, it is unclear what these weights are based on, or what formula is used to calculate the weighted scores. Due to the fact that this whitepaper and matrix were written by WSO2, which is a commercial API management platform provider, organizations are steered towards selecting their platform.

Gámez Gateway Comparison - As part of their work, which seeks to analyze the API Gateway paradigm and propose a SLA-Driven solution in an API Gateway design, Gámez Díaz et al. (2015) have compared API management features offered by various API gateway providers. This collection of providers consists of 13 Gateways including: *3Scale, Akana API Gateway, API Umbrella, Apiaxle, Apigee Edge, Axway API Gateway, Azure API Management, CA API Gateway, Mashape, Mashery API Gateway, Monarch API Manager, Repose* and *WSO2 API Management*. The aforementioned gateways were compared as based on a set of features such as *Security*, *Pricing plans support*, and *Lifecycle Control*. Confusingly, in their work, Gámez Díaz et al. (2015) use the terms 'API gateway' and 'platform' interchangeably. As a result, it is unclear whether the intention of the authors was to analyze features offered by the API gateway component, which is one of the main architectural components offered by the listed API management platform providers, or features provided by the platforms as a whole. Moreover, the set of features the aforementioned API gateways are compared on is very limited when compared to work on API management by authors such as De (2017).

Broadcom Playbook - In order to promote their API management platform, called Layer7, Broadcom has employed CA Technologies to compose an 'API Management Playbook' (CA Technologies, 2019). This playbook is targeted towards helping its readers comprehend the various reasons for API's importance in business, the API lifecycle and its relation to API management, the essential capabilities of an API management solution, and the features offered by the Layer7 platform. An evaluation method is presented which considers API management capabilities based on a collection of 13 use cases, or 'plays'. These use cases are broadly classified as having the goal of: API integration and creation, security, mobile and internet of things (IoT) development acceleration, and unlocking the value of data.

It is clear that even though this work presents an useful overview of API management capabilities, this overview directly matches the features offered by the Layer7 solution. As such, it may be concluded that the main purpose of this work is to convince and attract potential customers to Broadcom's platform. Furthermore, due to the commercial nature of this document, it can not be considered transparent in the sense that the source of the presented information is not known.

Accenture Advisory Report - In addition to their maturity model, Accenture has also published a consultancy report in 2019, advising banks on how to implement API management (Lees et al., 2019). Even though this report is specifically focused on the banking sector, and as a whole may thus be difficult to generalize and apply to other sectors, it contains several frameworks, figures and models that are related to this study.

The presented capability overview may be utilized by organizations seeking to implement API management processes, or may aid them in identifying API management platform capabilities that cater towards their needs. Furthermore, considering the aforementioned pillars are described in great detail and provide clear-cut guidelines for organizations to follow, this advisory report may be beneficial to organizations wishing to implement API management. However, when compared to the earlier described maturity models and frameworks, it may be difficult for organizations to self-assess their degree of maturity concerning API management.

Gartner Guidance Framework - Published by Gartner in 2019 (Gartner, 2019), this report is similar to those published by WSO2 and CA Technologies. It is aimed towards assisting technical professionals in selecting an appropriate API management platform. The general outline and structure of the framework is visible by reviewing the table of contents. Judging from this outline, it may be concluded that the framework is grounded in literature, using De (2017)'s work on API management as a foundation.

Devoteam Case Study - On their website, a Dutch company called Devoteam summarizes a case study in which the implementation of an API management platform at a large organization, *Liberty Global*, is described (Devoteam, 2016). As part of this case study, the case organization is described to have initially implemented an API gateway, which is argued to be an incomplete solution when compared to an API management platform. Furthermore, the organization required API management capabilities such as developer and partner onboarding, lifecycle management, documentation and testing, and analytics, which Devoteam (2016) does not consider to be capabilities that are typically provided by an API gateway.

In order to recommend an appropriate API management platform to the case company as based on their needs, an 'integration cookbook' was created, documenting the principles and guidelines for the usage of the API management platform and the different integration patterns. Unfortunately, similarly to the earlier described framework by Gartner, this cookbook is not publicly accessible. However, Devoteam (2016) mentions that it comprises policies that range from security policies, such as OAuth 2.0, OpenID Connect, basic authentication and IP whitelisting, to operational policies and monitoring and auditing policies. Judging from this information, it seems as though this cookbook primarily focuses on strategy, governance, and API management vendor evaluation and comparison. Due to its commercial nature however, it is unknown as to how this cookbook was created, whether it is able to be used to assess an organization's degree of maturity with regards to API management, or how complete it is.

A summarizing comparison matrix is presented in Ta-

Table 1
Comparison of existing frameworks and tools for API management assessment. It must be noted that the descriptions of the practices in each of these models typically had a low level of detail.

Framework	Type	Missing content	Goal	Availability	Grounding	Published	Transparent
Accenture API Management Suite	Maturity Model	Versioning, threat protection, authorization	Evaluation	Public	Industry	2014	No
Endjin Maturity Matrix	Maturity Matrix	Lifecycle management, dev. onboarding, traffic management, interface translation, monitoring	Evaluation	Public	Industry	2017	No
WSO2 Platform Evaluation	Whitepaper & Evaluation matrix	Dev. support	Platform selection	Public	Industry	2015	No
Gámez Gateway Comparison	Comparison Matrix	Dev. onboarding & support, authorization, traffic management, monitoring	Platform selection	Public	Literature	2015	Yes
Broadcom Playbook	Whitepaper	Version management, interface translation, logging	Consulting support	Public	Industry	2019	No
Accenture Advisory Report	Whitepaper	Dev. support, interface translation, logging	Knowledge sharing	Public	Industry	2019	No
Gartner Guidance Framework	Evaluation framework	Unknown	Platform selection	Commercial	Industry & Literature	2019	No
Devoteam Case Study	Case Study & Cookbook	Unknown	Platform selection	Commercial	Industry	2016	No

ble 1, which compares the artefacts. From the discussions of the different frameworks it becomes clear that the frameworks and tools either have a different goal (support consulting or platform selection), are not complete (missing capabilities), are not publicly available, or are not transparent in their construction. Our model combines the goal of evaluation and knowledge sharing, is public available, is transparent in the construction of the model, and describes the complete set of capabilities and practices for API management.

3. Research Approach

As discussed in the previous section, organizations that employ API management activities have no tools or frameworks at their disposal with which they may evaluate and improve upon their business processes regarding the topic of API management, that are publicly available, transparent, and grounded in both literature and industry. Despite growing interest in the topic of API Management in industry, more research is needed in order to fill knowledge gaps and identify best practices regarding the subject. Based on this problem statement the following research question is formulated, ensuring this research succeeds in achieving its goals. *How can organizations that expose their APIs to third parties evaluate their API management practices?*

Maturity models have been developed for organizations to use as an evaluative and comparative basis for improvement, in order to derive an informed approach for increasing the capability of a specific area within an organization (De Bruin et al., 2005). Moreover, maturity models have been designed to assess the maturity of a specific domain based on a set of criteria, and are a proven tool in the creation of collections of knowledge of practices and processes about a particular domain (Becker et al., 2009). Maturity models consist of a sequence of maturity levels for a class of objects, which typically include organizations or processes. The aforementioned sequence represents an anticipated, desired, or typical evolution path of these objects as discrete stages (Pöppelbuß and Röglinger, 2011). In order for an organization to progress along this path, criteria and characteristics relating to capabilities or process performance are included, which need to be fulfilled to reach a particular maturity level. The bottom stage of this path represents an initial state that may be characterized by an organization having little capabilities with regards to the domain under investigation. Organizations whose capabilities are of the highest maturity are located at the highest stage. Maturity refers to being well equipped to fulfill a purpose, i.e. having a higher level of sophistication, capability, or availability of specific characteristics (Mettler

et al., 2010). To appraise an organization's maturity, maturity models are commonly applied to assess the as-is-situation regarding the given criteria, so that improvement measures may be derived and prioritized (Iversen et al., 1999).

As De Bruin et al. (2005) and van Steenbergen et al. (2013) argue, the most well-known maturity model within the field of Information Systems is the Capability Maturity Model (CMM) from the Software Engineering Institute (Paulk et al., 1993). Studies have shown that since its inception, the CMM has inspired the development of hundreds of subsequent maturity models. These maturity models are aimed at a wide variety of domains and topics, including for example, project management (Crawford et al., 2007), corporate data quality management (Hüner et al., 2009), service integration (Arsanjani and Holley, 2006), and offshore sourcing (Carmel and Agarwal, 2006). Another example, which has been discussed in Section 2, is Accenture's API management maturity model (Tung, 2014).

An improvement over the maturity model is the focus area maturity model (FAMM) (Van Steenbergen et al., 2010; van Steenbergen et al., 2013). A FAMM allows a flexible number of of maturity levels and assesses the maturity per focus area which results in a more detailed evaluation. We follow the meta-model of FAMMs as presented by Jansen (2020) (shown in Figure 1), his work describes the development of a FAMM for the functional domain of software ecosystem governance. The functional domain is described by the set of focus areas that constitute it. Each focus area is composed out of a set of capabilities, which in the case of the API-m-FAMM are defined as the ability to achieve a goal related to API Management, through the execution of two or more interrelated practices. Combined, these practices and capabilities form the focus areas which describe the functional domain the topic of API management is composed of. A practice is defined as an action that has the express goal to improve, encourage, and manage the usage of APIs. Each individual practice is assigned to a maturity level within its respective capability. In order to establish an organization's degree of maturity with regards to the functional domain, the organization is asked to answer assessment questions linked to the capabilities the maturity matrix consists of. Based on the results of this maturity assessment, the organization is then guided towards incremental development of the domain, through a set of improvement actions with regards to the (missing) capabilities.

We apply the design methodology of Van Steenbergen et al. (2010) and De Bruin et al. (2005) in constructing our FAMM. The development of the FAMM is done in five phases: *Scope*, *Design*, *Populate*, *Test*, and *Deploy*. These phases are executed through a Systematic Literature Review (SLR), expert interviews, case studies, and numerous discussions among the authors. Every phase was concluded with the authors discussing the state of the model until consensus was reached on its contents and structure. This was done using online *Card Sorting* (Nielsen, 1995), with *Google Drawings* as a tool. In the *Scope*, *Design*, and *Populate* phases, the input for these discussions primarily consisted of the practices

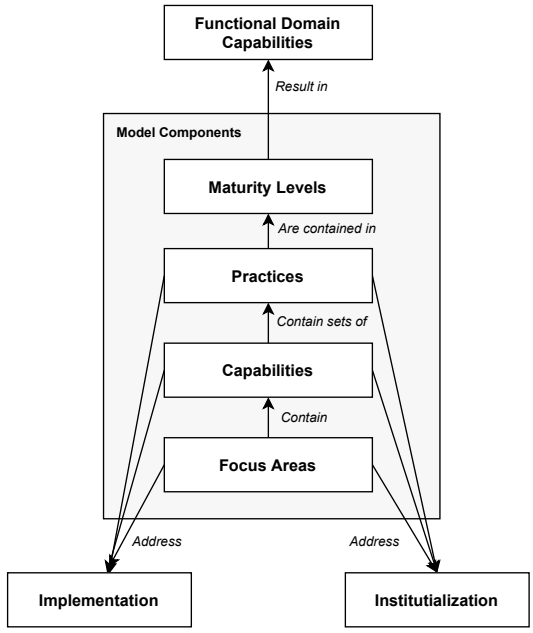

Figure 1: The meta-model of focus area maturity models as used in this research.

found in the different sources. The authors used card sorting to categorize these practices into capabilities, and to categorize the capabilities in areas. The various sources were used as inspiration for the discussions. During the *Test* phase the experts were also asked to comment on these categorizations. Comments that were deemed to be correct were then integrated into the model. Practices were assigned to a maturity level based on the ordering of practices within a capability and identified dependencies. In every phase we identified dependencies between the different practices. When a practice depends on a practice with maturity level l, the practice itself should have at least maturity level $l + 1$.

Figure 2 shows which methods were used in each phase, linked to the different intermediate versions of the API-m-FAMM. We use these versions to reference to a certain point in the construction of the model. A full description of the changes made during the different phases is available through the source data (Mathijssen et al., 2021). The source data also details the dependencies between the different practices. This document was published at different points: *v1 of the source data* corresponds with *v0.2*, *v2 of the source data* with *v0.3*, *v3 of the source data* with *v0.4*, and *v7 of the source data* with *v1.0*.

The initial model (*v0.1*) used the work of De (2017) as a starting point. Next a SLR (Mathijssen et al., 2020), based on the methodology developed by Okoli (2015) and guidelines composed by Kitchenham and Charters (2007), was used to further design and populate the model (resulting in version *v0.2*). In this SLR a comprehensive overview of literature related to API management was collected. This was accomplished by entering a series of relevant keywords in a list of scientific libraries, resulting in the extraction of an initial collection of 5152 books, research papers, theses and white

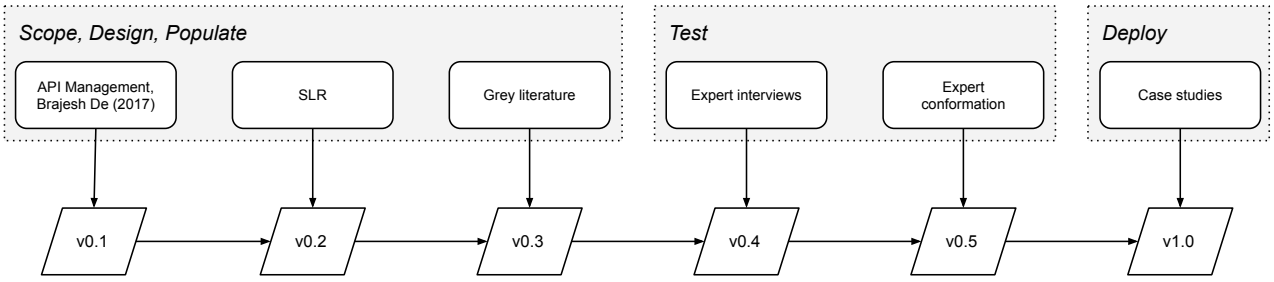

Figure 2: The steps executed in constructing, and the different intermediate versions of the API-m-FAMM.

papers. After having applied a set of inclusion and exclusion criteria, as well as removing duplicates, this collection was narrowed down to 43 published works. Next, the features API Management consists of were identified and extracted in the form of practices and capabilities from the complete body of literature. As a result of scanning and coding the body of included literature, 39 capabilities were identified and extracted. Among the 32 papers that were found to contain at least one practice or capability, 114 practices were identified and extracted.

To ground the model in both academic literature and industry experience two extra sources were used to construct the model. The collection of practices and capabilities resulting from the SLR are verified by using information gathered from grey literature, which includes white papers, online blog posts, websites, commercial API management platform documentation and third-party tooling (resulting in version *v0.3*).

Furthermore, experts on this topic are consulted to verify that the contents of the API-m-FAMM are complete and correct. To ensure that the selected experts are experienced and knowledgeable regarding the subject a purposive sampling technique is used, which refers to the deliberate choice of a participant due to the qualities the participant possesses (Etikan et al., 2016). The process of expert selection and interviews is visualized in Figure 3. Potential participants were identified and contacted through the usage of the partner network of AFAS Software (the company where the first and second authors are employed at). Additional potential participants were identified and contacted through the usage of the professional network of the authors. Furthermore, potential participants on both the API provider and consumer side are contacted. This ensures that knowledge stemming from both perspectives of API management is incorporated in the API-m-FAMM.

In order for the expert interviews to produce useful results, participants should be experienced and knowledgeable with regards to the topic of API management. As such, participants should adhere to the following requirements:

1. potential participants must indicate to be knowledgeable on a minimum of two out of the six focus areas of the the API-m-FAMM;
2. potential participants must have a minimum of 3 years of experience with either consuming, developing, integrating, providing, versioning, monitoring or managing APIs;
3. potential participants must be working at an organization as an architect, developer, engineer or product owner as part of a team working with APIs, or as a CTO, IT consultant or any comparable role.

In order to establish whether the potential participant satisfies the first requirement, a short preliminary survey is sent out, requesting the potential participant to indicate the degree of knowledge they possess with regards to the topic. 50 potential participants were contacted through e-mail. The survey was send together with a short list describing all the main elements that are contained in the API-m-FAMM, accompanied by a description (based on version v0.3). After having read these descriptions, potential participants are asked to denote their knowledge on the individual subjects through the use of 5-point Likert scale questions. In order to verify the second and third requirement, potential participants are asked to fill out their experience and current role within their organization. Then, if the potential participant passed all imposed requirements for participation, participants were asked to read an information sheet and sign an informed consent form.

The sampling process resulted in the selection of nine experts, whom are presented in Table 2 (two respondents were excluded based on the knowledge assessment). This ratio may seem high at first glance, but considering only one or two focus areas were discussed during an expert interview, it was deemed to be unnecessary for participants to be knowledgeable on more than two focus areas as only those particular focus areas were included for discussion during interviews. Based on an interviewee's knowledge regarding the six focus areas the API-m-FAMM consists of, one or more focus areas are selected for evaluation. Focus areas were evaluated through eleven interviews, which resulted in each focus area being evaluated three times. These interviews were subsequently transcribed and processed to incorporate all comments into the FAMM.

During the interviews, which are semi-structured in nature, the API-m-FAMM in its entirety is first presented to the

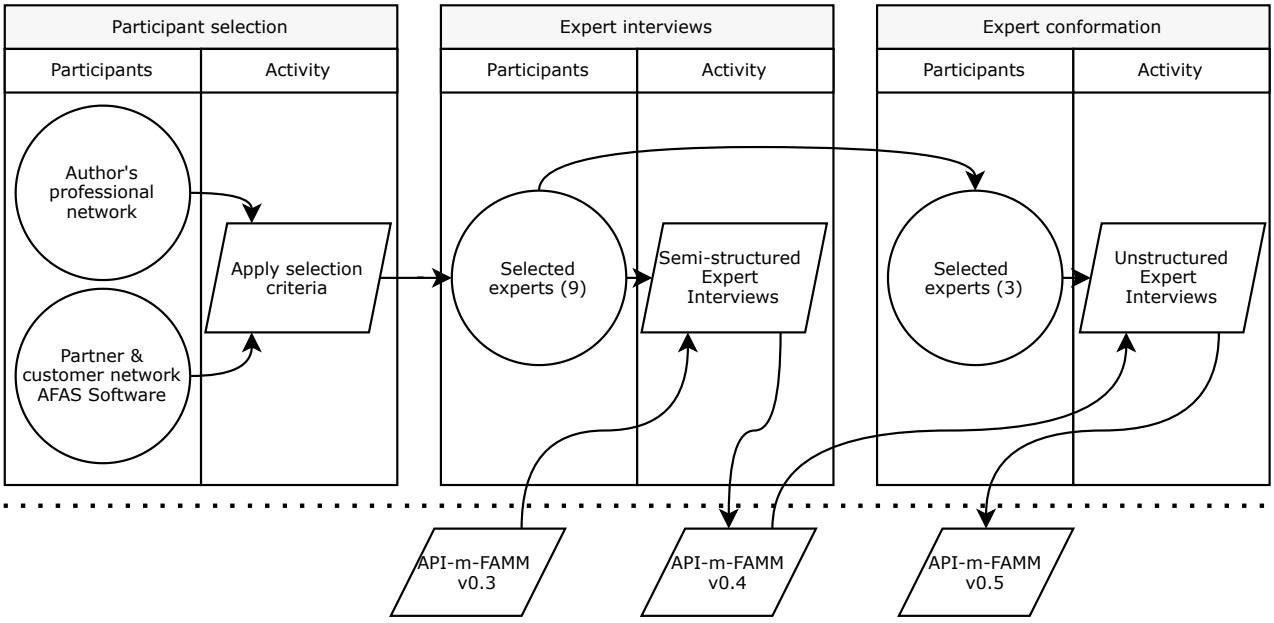

Figure 3: The process of expert selection explained. First we selected possible experts based on the author's professional network and the case company's partner and customer network. While 50 experts were invited, eleven experts responded to the survey. We selected nine experts by applying our selection criteria and interviewed them using on *v0.3* of the model. The results of these interviews were integrated in *v0.4*. From the nine experts we selected three experts to conduct extra conformation interviews. The results of these interviews were integrated in *v0.5*.

Table 2
Interviewees and the current role they fulfill within their organization, as well as the years of experience they have with either consuming, developing, integrating, providing,versioning, monitoring or managing APIs. Additionally, it is shown which focus areas were discussed with interviewees, as well as the duration of the interviews.

Interviewee	Experience	Hours	Community	Security	Lifecycle	Monitoring	Performance	Commercial
$API\ Evangelist$	10+	1.0	✓					✓
CEO_A	10+	1.5	✓					
CEO_B	10+	5.5		✓	✓	✓		
$Engineer$	10+	1.0				✓	✓	
$IT\ Consultant$	10+	3.5	✓	✓	✓			✓
$Product\ Manager$	6	3.0		✓				
$Lead\ Engineer_A$	10+	1.0				✓	✓	
$Lead\ Engineer_B$	10+	1.0					✓	✓
$Lead\ Engineer_C$	5	1.5			✓			
Total	N/A	19	3	3	3	3	3	3

expert. Next, the focus area that is selected for evaluation and each capability it comprises is described. Then, all practices these capabilities consist of are elaborated upon. For each capability, experts are asked whether they are familiar with it and whether they believe it is assigned to the correct focus area. Similarly, experts are asked whether they are familiar with each practice, and whether they believe it is assigned to the correct capability. Additionally, they are asked whether they can identify any dependencies with regards to the implementation of other practices. After having answered these questions for a capability and the practices it comprises, experts are asked to rank practices in terms of their perceived maturity and complexity for each capability. This ranking exercise is performed by using the same card sorting technique that was used to initially structure the API-m-FAMM (via *Google Drawings*). Figure 4 shows an example ranking result of a single capability. These results are analyzed and combined with any implementation dependencies to create a final ranking of the practices (resulting in *v0.4* of the FAMM).

The resulting version of the FAMM was evaluated through a second cycle of unstructured interviews with three experts originating from the same sample of experts that were inter-

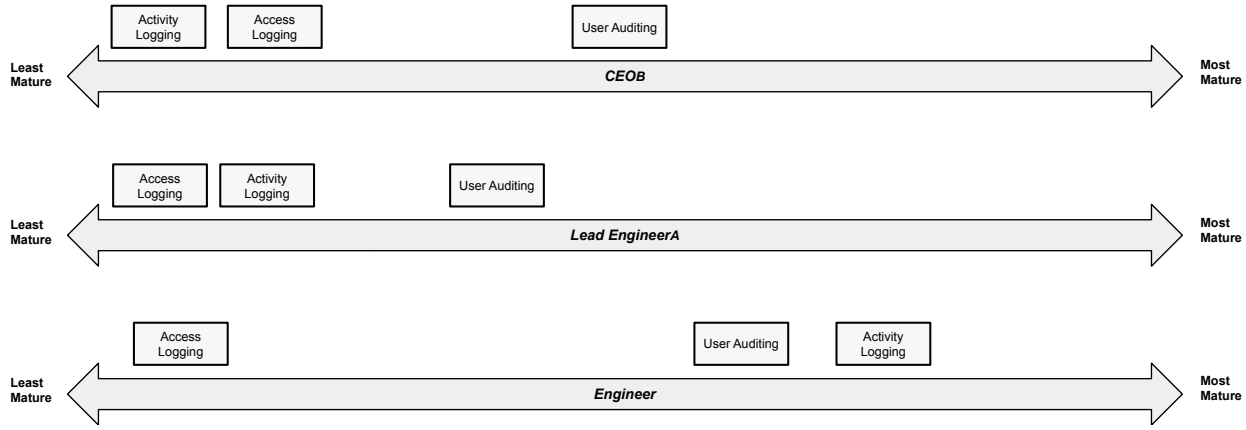

Figure 4: Ranking of practices in the capability *Logging* (intermediate practices are shown as during the interviews) as done by the experts.

viewed during the first evaluation cycle: $Product\ Manager$, $IT\ Consultant$, and $Lead\ Engineer_A$. These experts are knowledgeable on a large portion of areas and based on that criteria were asked to participate in the second evaluation cycle. During these interviews the changes made to the FAMM were discussed and confirmed (resulting in *v0.5* of the FAMM).

Finally the model was deployed in four companies, by evaluating five different software products. As a result of the case studies the model was changed by removing one practice and improving the descriptions of three other practices. This resulted in *v1.0* of the API-m-FAMM. The contents of the API-m-FAMM are discussed in Section 4, and the case studies are described in Section 5.

4. The API Management Focus Area Maturity Model

The previous section motivates the usage of the focus area maturity model as an artifact to capture the functional domain of API management, and describes the methods used in constructing it. This section introduces the **API m**anagement **F**ocus **A**rea **M**aturity Model and its contents: the **API-m-FAMM**.

The scope of the API-m-FAMM is the domain of API management: the API-m-FAMM aims to support organizations that expose their API(s) to third-party developers in their API management activities. Based on the SLR API management is defined as *an activity that enables organizations to design, publish and deploy their APIs for (external) developers to consume*. However, three adjustments are made to the scope of the API-m-FAMM with respect to this definition.

First, considering that a prerequisite for performing API management as an activity, is for an organization to already have designed and created an API, the actual design and creation process of APIs itself is excluded from the API-m-FAMM. These processes overlap with software engineering in general and would require the inclusion of capabilities such as agile software development and test-driven development. In contrast however, the publication, maintenance, and deprecation (which are contained in the API lifecycle) of APIs are included in the scope of the API-m-FAMM, considering that these are concerned with capabilities such as lifecycle management, developer enablement, security, and analytics.

Second, the management of internal APIs is not explicitly targeted with the the API-m-FAMM, in contrast to management of partner and public APIs. Capabilities such as developer enablement, security and analytics are of lesser importance with regards to managing internal APIs, considering that these APIs are exclusively used for internal app integration and development. However, the case studies (discussed in Section 5 show that the API-m-FAMM still proves to be useful for such organizations wishing to incrementally improve upon capabilities such as lifecycle management, as well as increasing internal performance of their API(s).

Finally, the API-m-FAMM seeks to provide practitioners with incremental capability improvements that are tool, technology, and platform-independent. For example, there are many tools which organizations may utilize to implement monitoring, communication, and support capabilities. In the same vein, capabilities regarding traffic management or security, such as authentication, authorization and threat protection, may be relatively straightforward to implement through the use of commercially available gateway or management platform solutions. However, some organizations may opt to develop these solutions in-house, or not wish to use such solutions for a variety of reasons. To ensure that the API-m-FAMM is generalized to both these types of organizations, comparisons between specific tools or management platform solutions are excluded from the scope of the model. An example of practices that were removed because of this is *Visual Data Mapping*, which is exclusively provided by the *Axway* API management platform [1] and *Error Handling*, which is implementable through the use of the *Apigee* plat-

[1] https://www.axway.com/en/products/api-management

form [2]. Another example is the practice *Standardized Authorization Protocol*, which was initially included as *OAuth 2.0 Authorization*, but was renamed with a referral to the OAuth 2.0 protocol being included in its description instead.

As discussed by De Bruin et al. (2005), the design phase explains how the needs of the intended audience are met through answering the *why*, *how*, *who*, and *what* question. **The 'why'** for the API-m-FAMM is that its main goal is to assist organizations that (plan to) expose their API(s) to third-party developers to assess and evaluate their degree of maturity with regards to their API management capabilities. **The 'how'** is that organizations can utilize the API-m-FAMM to assess their as-is situation with regards to their API management capabilities, and then subsequently incrementally improve upon these capabilities by implementing practices that are of a higher maturity. **The 'who'** involved in applying the API-m-FAMM may vary across organizations, depending on characteristics such as their size in terms of employees involved in the API program, number of exposed APIs, and the degree of incoming traffic. For example, for a small organization that exposes one, mildly popular API, it is likely that a small number of employees are familiar with the API program and the activities involved in managing it. These employees may then utilize the API-m-FAMM to assess the as-is situation, and then use the model as a road map to incrementally implement capabilities and practices to reach a higher level of maturity. In contrast, a large organization that exposes multiple, popular APIs that generate large loads of traffic, is likely to employ multiple development teams, product owners and designers whom are involved with the API program. In this case, it is unlikely that a solitary employee or a small group of employees will be able to assess the current as-is situation of the API program. Instead, information for this assessment will have to be extracted through meetings with employees from varying teams and backgrounds whom are involved in the various aspects of API management, such as community engagement, security measures, monitoring capabilities, and lifecycle management. Alternatively, consultants with a thorough understanding of API management and the API-m-FAMM may be able to apply the model by conducting interviews with all relevant stakeholders involved in the organization's API program. **The 'what'** that can be achieved through the application of the API-m-FAMM is an insight into the current maturity of an organization with regards to its API management capabilities, as well as a path to incremental implementation and improvement of more mature, specific practices.

The API-m-FAMM, based on the SLR, the expert interviews, and case studies, is presented in Figure 5. Detailed descriptions, preconditions, and reference literature can be found in the source data Mathijssen et al. (2021). The FAMM consists of six focus areas, 20 capabilities, and 80 practices. The highest maturity level is 10.

During the expert interviews, part of the *Test phase* (see Figure 2), an interim version of the API-m-FAMM was measured on four criteria using likert scale questions:

- **Operational Feasibility**: How likely do you think it would be that an organization would actually use the API-m-FAMM in practice to evaluate and improve upon their API management related processes?
- **Ease of Use**: How easy do you think it would be to understand the API-m-FAMM's content and use it to self-assess and evaluate your organization's maturity in API management?
- **Usefulness**: How useful do you think the API-m-FAMM would be in providing you and your organization with valuable and interesting insights in your organization's API management related processes?
- **Effectiveness**: How effective do you think the API-m-FAMM would be in helping you and your organization improve on their API management related processes?

The results of this interim evaluation are listed in Table 3. These results should be prefaced by a few initial remarks. Firstly, it should be noted that during the interviews, an intermediate and unfinished version of the API-m-FAMM was presented to experts. One of the implications of this is that maturity levels were absent from this version of the model. Because of this, experts often found it difficult to envision what the final version would look like, as well as to properly judge its ultimate capabilities and potential for helping an organization in improving their API management maturity. Additionally, only one or two of the six focus areas the model consists of were selected for discussion, which, in some instances, impaired experts' ability to grasp the API-m-FAMM's scope as a whole on a conceptual level. Furthermore, the descriptions given for the focus areas, capabilities, and practices during the interview were summarized and shortened. Lastly, the experts had not familiarized themselves with the contents of the API-m-FAMM prior to the interview taking place.

Still the experts generally responded positively to the model, and expressed interest in using it in practice to evaluate their organization's API management related processes and assess their API management maturity. For example, *Product Manager* concluded the interview with the following remark: *"I think this - the API-m-FAMM - is a very thorough analysis. You have made a very nice overview that can help organizations with deciding what and when they have to do when wanting to start with an API. If they want to bring something to the market quickly, this helps them realize they must first have implemented the processes on the lower levels, and have to start small. Actually, for many organizations that already have an API or want to start building one, this is the roadmap they should follow for a good API strategy."* The evaluation criteria as stated earlier should be interpreted as how the expert thinks the final model will score. These scores are no proof of the actual effectiveness or usefulness of the model, but an indication of how the model will be received by industry in the future. In this phase the criteria are used as *marketing research*: how likely will the API-m-FAMM succeed in industry.

[2] https://cloud.google.com/apigee/api-management?hl=nl

Components \ Maturity Levels		0	1	2	3	4	5	6	7	8	9	10
1.	**Lifecycle Management**											
1.1.	Version Management			Implement Evolutionary API Strategy			Implement Multiple API Versions Strategy	Implement API Deprecation Protocol	Check Backwards Compatibility			
1.2.	Decoupling API & Application		Decouple API & Software Versioning			Decouple Internal & External Data Model	Decouple Internal & External Data Format	Decouple Internal & External Transport Protocol				
1.3.	Update Notification			Distribute Changelogs	Distribute Versioning Notification through Channel(s)		Extend API with Versioning Information				Announce API Versioning Roadmap	
2.	**Security**											
2.1.	Authentication		Implement Basic Authentication			Implement Authentication Protocol			Implement Single Sign-on			
2.2.	Authorization			Implement Access Control		Implement Token Management		Implement Standardized Authorization Protocol	Implement Authorization Scopes			
2.3.	Threat Detection & Protection		Implement Allow & Deny IP Address Lists	Implement Injection Threat Protection Policies			Implement DoS Protection		Implement Security Breach Protocol		Conduct Security Review	Implement Zero Trust Network Access
2.4.	Encryption		Implement Transport Layer Encryption		Implement Certificate Management							
3.	**Performance**											
3.1.	Resource management			Implement Load Balancing			Implement Scaling	Implement Failover Policies				Implement Predictive Scaling
3.2.	Traffic Management		Set Timeout Policies	Implement Request Caching	Perform Request Rate Limiting	Perform Request Rate Throttling	Manage Quota	Apply Data Volume Limits			Prioritize Traffic	
4.	**Observability**											
4.1.	Monitoring		Monitor API Health		Monitor API Performance		Monitor Resource Usage					
4.2.	Logging		Log Errors	Log Access Attempts	Log Activity		Audit User Activity					
4.3	Analytics			Report Errors	Broadcast API Status			Generate Custom Analysis Reports	Set Alerts		Enable Predictive Analytics	
5.	**Community**											
5.1.	Developer Onboarding		Facilitate Developer Registration			Provide SDK Support	Implement Interactive API Console			Provide Sandbox Environment		
5.2.	Support		Establish Communication Channel			Manage Support Issues		Dedicate Developer Support Team				
5.3.	Documentation		Use Standard for Reference Documentation		Provide Start-up Documentation Including Samples		Create Video Tutorials					
5.4.	Community Engagement		Maintain Social Media Presence		Provide Community Forum	Provide Developer Portal			Organize Events		Dedicate API Evangelist	
5.5.	Portfolio Management		Enable API Discovery			Provide API Catalog	Bundle APIs					
6.	**Commercial**											
6.1.	Service-Level Agreements		Publish Informal SLA		Provide SLA			Monitor SLA Proactively	Customize Personalized SLA			
6.2.	Monetization Strategy							Adopt Subscription-Based Monetization Model		Adopt Tier-Based Monetization Model	Adopt Freemium Monetization Model	Adopt Metering-Based Monetization Model
6.3..	Account Management			Implement Subscription Management System					Report on API Program Business Value	Provide Subscription Report to Customer	Proactively Suggest Optimizations to Customer	

Figure 5: The API-m-FAMM: a focus area maturity model for API management. Consisting of six focus areas, 20 capabilities, and 80 practices.

5. Case Studies

The case studies, as described in Section 3, are evaluative in nature and are aimed at determining to what degree the API-m-FAMM succeeds in aiding an organization in evaluating and improving upon their API management related business processes in practice. This corresponds with the *Deploy* phase of the design of the maturity model. The API-m-FAMM is evaluated with an embedded case study at the company the first and second author are employed at, and through case studies at four different companies. Data resulting from the application of the API-m-FAMM is collected through an Excel spreadsheet. Finally, participants evaluated the API-m-FAMM on the same criteria that were employed as part of the first evaluation cycle. These processes are part of the case study protocol that is used in conducting the case

Table 3
The rankings given by the experts in response to the questions corresponding to the four evaluation criteria, as well as their averages and standard deviation.

Expert	Operational Feasibility	Ease of Use	Usefulness	Effectiveness
API Evangelist	3	2	4	3
CEO_A	4	3	4	4
CEO_B	3	2	3	4
Engineer	4	4	4	5
IT Consultant	2	2	3	4
Product Manager	4	4	3	3
$Lead\ Engineer_A$	4	3	5	4
$Lead\ Engineer_B$	3	2	4	3
$Lead\ Engineer_C$	4	3	5	4
Average	3.4	2.8	3.9	3.8
Std. Dev.	0.68	0.62	0.74	0.63

studies.

The main objective of the case study is to determine to what degree the API-m-FAMM succeeds in aiding an organization in evaluating and improving upon their API management related business processes in practice. The selected organizations were provided with a visual copy of the API-m-FAMM, as well as a copy of the source document describing the focus areas, capabilities, and practices the model consists of in detail. Considering that the API-m-FAMM was previously presented to selected participants as part of the first evaluation cycle, they were already informed with regards to the focus of this research as well as the purpose and structure of the API-m-FAMM. After having familiarized themselves with the contents of the API-m-FAMM, participants are asked to asses their maturity by filling out whether each practice has either been or is:

- **Implemented**: the practice has currently been implemented in the organization.
- **Implementable**: the practice has not been implemented, but in theory is implementable. Depending on the organization's needs and plans, the practice will either be implemented in the short-term, long-term, or not at all.
- **Not applicable**: the practice has not been implemented, and is not applicable as it will most likely never be implemented. This may be caused by a number of reasons. For example, the practice may not be of added value, or not desirable for the organization to implement because it does not align with the organization's goals, vision, or needs.

Lastly, participants are asked to fill out a short survey consisting of a series of questions that are similar to those that were asked as part of the expert interviews during the first evaluation cycle. The main difference is that during the expert interviews, the questions were phrased the future tense due to the API-m-FAMM not being completed yet, while during the case studies the questions were posed in the past tense. The purpose of these questions is for the participants of the case study to evaluate the API-m-FAMM with regards to its *operational feasibility*, *ease of use*, *usefulness*, and *effectiveness*. These questions are aimed at determining the degree to which the the API-m-FAMM has succeeded in aiding the participating organization in evaluating and improving upon their API management related business processes in practice.

For the embedded single-case study, the API-m-FAMM is applied to two software products that are developed by AFAS Software. Within this case company, the API-m-FAMM was deployed and evaluated with two development teams that are working on two separate software products. Afterwards the evaluation was discussed with the teams as well. We discuss these two products and the highlights from the assesment.

AFAS Software is a Dutch vendor of ERP software based in Leusden, the Netherlands. Additionally, AFAS has offices in Belgium and Curaçao. The privately held company currently employs over 500 people and generated €191 million of revenue in last year (2020).

AFAS Profit - AFAS' main software product is Profit, which is an ERP package consisting of different modules such as Fiscal, Financial, HRM, Logistics, Payroll, and CRM. Currently, this product has over 2 million users across over 11.000 small, medium and large organizations. AFAS Profit provides customers with two APIs: a REST API and a SOAP API. Both of these APIs offer the same functionalities, and customers may decide on using either of these depending on their preferences. These APIs are called about 500 million times a month. Furthermore, standard connections with external software products and applications that utilize these connectors and that AFAS is partnered with are offered through AFAS' partner portal.

With regards to *Lifecycle Management* the interviewees have marked *Implement Multiple API Versioning Strategy* as not applicable. During the discussion they explained that this is due to the fact that the version of the connectors are directly tied to the version of the application itself. However, in the event of changes, the consumers had to be notified and plan the required changes. In order to notify consumers of updates, Profit publishes release notes describing general updates made to the product as well as specific updates made to the connectors.

Only a few practices from the *Commercial* focus area have been implemented. Consumers are to adhere to a SLA, which contains agreements on fair use, time-out policies, and uptime guarantees. Furthermore, no strategy for monetizing

the APIs is employed considering that this is already done indirectly through the product's licensing.

AFAS Focus - Currently, AFAS is developing a new version of its ERP software, which is called Focus. This product shares nothing in terms of the technology and codebase used with AFAS Profit, and is cloud-based as well as generated by using the ontological model of an enterprise as input. Considering that since the time development first commenced some modules such as Financial have been developed, AFAS is currently in the process of transferring customers from the current Profit product to Focus. AFAS Focus only supports REST APIs, which support both the XML and JSON data formats. The endpoints are described using the OpenAPI specification and make use of the OAuth application token flow for authentication. The current size and state of AFAS Focus encompasses about twenty available endpoints, which are directed at a few specific integrations. In the future this number will increase as Focus continues to grow and branch out to more partners.

As AFAS Focus is not yet mature, a number of areas are not yet relevant. While groundwork for *Community* is done, there is not yet a community that needs to be supported. This also results in the underdevelopment of the *Commercial* area. The number of customers do not yet pose strong demands on *Performance*, resulting in a number of advanced practices not being implemented.

The remaining case studies are conducted with multiple organizations. These organizations were partly selected by contacting the experts that were previously interviewed as part of the first evaluation cycle. Furthermore, other organizations were selected through the utilization of the network of the authors. Please note that the names of some of these organizations are anonymized. Explicit consent to include the name of the organization was obtained of those that are not.

ConsultComp - *ConsultComp* is a multinational professional services network, is one of the Big Four accounting organisations, and is among the largest professional services networks in the world by revenue and number of professionals. *ConsultComp* provides audit, consulting, financial advisory, risk advisory, tax, and legal services with over 300.000 professionals globally. Aside from these services, the organization also develops software products for customers, which are developed in-house in their office in the Netherlands. The team involved in developing these products utilizes internal APIs, third-party service integrations to access data from service providers, and also is in the process of starting to expose (partner) APIs to customers.

In the scope of the *Lifecycle Management* focus area, the internal API is versioned using the *evolutionary versioning strategy*. Considering that the organization's API is exclusively used internally, practices corresponding to *Update Notification* are not implemented. Noticeably, a large part of the practices corresponding to the *Security* focus area have been implemented. In order to authenticate internal consumers of the API, an authentication protocol along with the SSO method is used. Furthermore, all practices belonging to *Threat Detection & Protection* and *Encryption* have been implemented to further secure the product. Virtually all practices belonging to the *Community* and *Commercial* focus areas have been marked as not implemented or not applicable. Similarly to capabilities such as *Update Notification* and parts of the *Traffic Management* and *Analytics*, this is logical considering that in the current case of an internal API, there simply is no community surrounding it to manage as of yet.

Exact - Exact is a multinational organization that provides ERP software and was founded in the Netherlands. Apart from their headquarters in the Dutch city Delft, Exact also has offices in 20 other countries, and currently employs over 1850 people and annually generates €209 million of revenue. Exact provides customers with various products, such as an integrated ERP package, and a package that incorporates modules that are targeted towards CRM, HRM, and workflow management. Exact's main software product that is offered in the Netherlands and Belgium is called Exact Online, which is a package consisting of modules such as accountancy, CRM, and project management. This SaaS product currently has over 500.000 users and is fully internet-based. Exact Online provides customers with two main API types: a REST API and a XML API. These APIs comprise a range of endpoints which combined are called about 700 million times a month.

All practices belonging to the *Lifecycle Management* focus area have been marked as *Implemented*. Both versioning strategies are marked as implemented, as well as the deprecation protocol and backwards compatibility checking practices. The product is also mature with regards to update notification, with the exception of announcing an API versioning schedule. Similarly, most practices in the *Security* focus area have been marked as implemented, including conducting security reviews: over the course of the past few years, Exact has conducted security audits at 2000 organizations. Furthermore, Exact's Zero Trust Network Architecture is implemented through a third party platform. Consumers of Exact Online are provided with an extensive SLA, which contains elements such as uptime guarantees, fair use policies, and agreements on rate and data limiting. Furthermore, while access to the product as a whole is monetized through monthly licensing fees, no monetization model that specifically and exclusively applies to the APIs is used.

Uber - Uber is a large-scale multinational technology organization that was founded in the United States. It provides multiple services, such as ride-hailing, food delivery (Uber Eats), and package delivery. Uber is estimated to have over 93 million monthly active users worldwide. In 2012, Uber established its international headquarters in Amsterdam, the Netherlands. Here, among other things, development teams are responsible for optimizing the performance and scalability of the public API that is provided by the organization, as well as developing new functionalities.

In terms of the *Lifecycle Management* focus area, nearly all practices have been implemented. The organization is able to version their APIs using the evolutionary strategy,

as well as the multiple API versioning strategy. To this end, Uber also has a deprecation protocol and backwards compatibility checking methods in place. Furthermore, mechanisms to provide consumers with information regarding updates to the API are in place, with the *announce versioning schedule* practice being planned for implementation in the foreseeable future. Most practices have been implemented for the *Performance* focus area. Considering that Uber is a large-scale multinational organization, this is to be expected. Regarding *Resource Management*, the organization has implemented advanced scaling methods, considering that automatic scaling takes place when certain resource thresholds are exceeded. Furthermore, Uber has runbooks in place that detail what course of action should be taken when the scaling strategy fails. Additionally, considering that the organization has various data centers that are located in different continents and countries, there are *failover policies* in place which allow the organization to mitigate outages.

The overall results of the case studies are presented in Table 4, showing the amount of practices that are *implemented* for each capability. Consequently, this means that the other implementation categories (*implementable*, and *not applicable*) are not taken into account in this overview. Furthermore, before discussing these results, it should be noted that the percentages representing the average amount of practices that are implemented in each focus area should be interpreted in an indicative manner, as some practices may encompass a much larger amount of work than others. Moreover, some case companies vary heavily in terms of their vision, size, usage of APIs, and goals. For example: Uber, AFAS Profit and Exact may be roughly classified as large and mature organizations that have been exposing public or partner APIs for a long period of time, which is reflected by the observation that these organizations have implemented the majority of practices for most focus areas. *ConsultComp* is also a large organization in size, but the product that was evaluated has only been utilizing internal APIs for a short period of time. Hence, some focus areas are largely irrelevant within the scope of this product. Similarly, the AFAS Focus product has been utilizing partner APIs for a relatively short period of time and is currently in the process of expanding the services it provides as well as the assets it exposes. Due to the current scale of the product and the stage of development it is in, some practices are currently not yet necessary to be implemented.

Firstly, it was observed that some practices are mutually exclusive. This phenomenon may be observed in the *Authentication* capability, where in some cases the *Implement Basic Authentication* has been marked as *not applicable* when the *Implement Authentication Protocol* had been marked as *implemented*. This is indeed a logical result, considering that only one method of authentication is necessary. Similarly, such choices are also observed between the *Implement Interactive API Console* and *Provide Sandbox Environment Support* practices.

Secondly, it may be observed that the mature organizations (Uber, Exact and AFAS Profit) as a whole have implemented the majority of practices across most focus areas. This is particularly the case for *Lifecycle Management*, *Security*, *Performance*, *Observability* and *Community*. However, the AFAS Profit product forms an exception to this observation, considering the relatively high amount of practices that are *not applicable* and *implementable*. The practitioners noted that practices that have been marked as such have been discussed internally in the past, and were concluded to not be desirable to be implemented due to them not aligning with the prevalent vision and goal of the product.

Thirdly, we observe that within the earlier mentioned focus areas, some practices on the higher end in terms of maturity are often not implemented across the board, such as *Announce Versioning Roadmap*, *Implement Predictive Scaling*, and *Prioritize Traffic*. This is supportive evidence for these practices having high maturity levels. Furthermore, in most cases such practices were marked as *implementable*, signalling that organizations that have done so are interested in implementing these practices in the future. It should be noted however, that the time-span in which organizations plan on implementing such practices may vary greatly.

Lastly, a common trend that may be observed across all organizations is that the *Commercial* focus area is underdeveloped. Particularly, this is the case for the *Monetization Strategy* capability, considering that no organization has marked any of these practices as *implemented*. However, the practitioners at Exact have expressed interest in implementing a monetization strategy in the future. Specifically, the practitioner at Exact mentioned that he expects that monetization for APIs will become increasingly more common in the future, and that he suspects that large-scale organizations that expose high-traffic public APIs already monetize their APIs. However, no concrete evidence of this has been found as part of this case study and its participants.

The practitioners experienced the usage of the API-m-FAMM in a positive manner. As can be seen in Table 5, the *ease of use*, *usefulness*, and *effectiveness*, were all ranked with an average score of 4 or higher. This supports the usefulness and effectiveness of the API-m-FAMM in achieving its goal of aiding organizations in assessing their current maturity in API management and guiding them towards achieving higher levels of maturity. However, the average score attributed to the API-m-FAMM's operational feasibility is notably lower when compared to the other criteria. In particular, the scores given by practitioners corresponding to the AFAS Focus and Profit products are among the lowest. This may be explained by the fact that the AFAS Profit product is almost fully matured and developed, which reduces the incentive among its practitioners to repeatedly consult the API-m-FAMM for guidance. This is further compounded due to the practitioners already being aware of most practices that have not yet been implemented, as well as having previously discussed them internally in the past. For the case of AFAS Focus, this may be explained by the observation that a large portion of focus areas and capabilities are irrelevant for the product given its current development stage, which in turn reduces

Table 4
The results of the deployment of the API-m-FAMM at the case companies. For each capability, the amount of implemented practices is shown. The percentages indicate the share of practices that have been implemented out of the total amount of practices that are assigned to the focus area. Please note that these percentages should be interpreted in an indicative manner, as the practices are not weighted, and some practices are more extensive than others.

	Focus Area	AFAS Focus	AFAS Profit	*ConsultComp*	Exact Online	Uber	Avg	Stdev
1	**Lifecycle Management**	58%	50%	42%	92%	83%	65%	
1.1	Version Management	2	2	1	4	4	2.6	1.20
1.2	Decoupling API & Application	4	2	4	4	3	3.4	0.80
1.3	Update Notification	1	2	0	3	3	1.8	1.17
2	**Security**	53%	73%	80%	87%	100%	79%	
2.1	Authentication	1	1	2	2	3	1.8	0.75
2.2	Authorization	2	2	2	4	4	2.6	0.98
2.3	Threat Detection & Protection	3	6	6	6	6	5.4	1.20
2.4	Encryption	2	2	2	1	2	1.8	0.40
3	**Performance**	36%	45%	55%	64%	82%	56%	
3.1	Resource Management	3	3	3	2	3	2.8	0.40
3.2	Traffic Management	1	2	3	5	6	3.4	1.86
4	**Observability**	75%	67%	67%	92%	83%	77%	
4.1	Monitoring	2	3	3	3	3	2.8	0.40
4.2	Logging	4	3	4	4	4	3.8	0.40
4.3	Analytics	3	2	1	4	3	2.6	1.02
5	**Community**	28%	78%	6%	94%	78%	57%	
5.1	Developer Onboarding	2	1	0	4	3	2.0	1.41
5.2	Support	2	3	1	3	3	2.4	0.80
5.3	Documentation	1	2	0	3	2	1.6	1.02
5.4	Community Engagement	0	5	0	5	3	2.6	2.24
5.5	Portfolio Management	0	3	0	2	3	1.6	1.36
6	**Commercial**	0%	42%	0%	33%	50%	25%	
6.1	Service-Level Agreements	0	2	0	2	4	1.6	1.50
6.2	Monetization Strategy	0	0	0	0	0	0.0	0.0
6.3	Account Management	0	3	0	2	2	1.4	1.20

the effectiveness of the API-m-FAMM in this case.

Interestingly, the API-m-FAMM was ranked fairly high by *ConsultComp*'s practitioner, regardless of the fact that, similarly to the AFAS Focus product, some focus areas and capabilities do not apply to the product given its utilization of internal APIs. However, *ConsultComp*'s intent of developing partner APIs in the future may provide an explanation for the practitioner's appreciation of the API-m-FAMM. Regarding the average score given to the *operational feasibility* criterion (Table 5), the researchers suspect that this is on the low end due to a lack of incentive for practitioners to re-use the API-m-FAMM after the initial assessment.

6. Discussion

The API-m-FAMM has shown to be an effective tool to measure and asses the maturity of an organization with regard to API management. The case studies participants rank the model with an average of 4 out 5 for ease of use, usefulness, and effectiveness. In comparison with the ranking of the experts, made during the interviews, only operational feasibility is ranked lower with a 3.2 out of 5 (shown in Table 5). This last one is explainable, as two out of five participants remarked that there is no added value of a continuous evaluation with the API-m-FAMM. The rankings given by the experts during the development of the model were a bit lower for usefulness and effectiveness, while being more than a

Table 5
The rankings given by the case companies' practitioners in response to the questions corresponding to the four evaluation criteria as well as their averages. Included are the average and standard deviation from Table 3 for reference.

Product	Operational Feasibility	Ease of Use	Usefulness	Effectiveness
AFAS Profit	2	4	4	3.5
AFAS Focus	2	3	4	3
ConsultComp	4	4	3	4
Exact Online	4	5	5	4
Uber	3	4	4	5
Average	3.2	4.0	4.0	3.9
Std. Dev.	0.89	0.63	0.63	0.6
Expert Interviews Average	*3.4*	*2.8*	*3.9*	*3.8*
Expert Interviews Std. Dev.	*0.68*	*0.62*	*0.74*	*0.63*

point lower for ease of use. This last criteria is important for the general adaption of the API-m-FAMM and maturity models in general.

Experts occasionally suggested during interviews that they are of the opinion that a bigger range of practitioners could be reached if the accessibility of the API-m-FAMM as a tool would be improved. One way in which this could be achieved, is by developing a web-app through which practitioners may easily navigate, as well as read focus area, capability, and practice descriptions, and then mark which practices are or are not implemented within their organization. In doing so, the Excel spreadsheet and source document that were used as part of the case studies could be combined into a single easily usable tool. Currently, the API-m-FAMM is maintained on `maturitymodels.org`. While this website offers a visual alternative to the source document considering that descriptions are included, it does suffer from some limitations. Most notably practitioners are unable to use the website to perform an evaluation and denote whether practices have been implemented. Therefore, we published a *do-it-yourself* kit that was used during the case studies. Improving this would allow practitioners to share the current as-is situation of their organization's API management maturity with management and stakeholders. These results could be used as a benchmark for other practitioners through an opt-in consent. The opportunity for improved visualization of results extracted from FAMMs was also previously identified by Spruit and Röling (2014). In this work, it is suggested that effective result visualization may be accomplished through spider charts, since most managers are familiar with such a type of representation.

Another opportunity lies in the potential to customize and adapt the API-m-FAMM depending on certain organizational characteristics and goals. For example, the case studies have shown that certain focus areas are irrelevant for organizations that exclusively utilize internal APIs. For instance, this is visible in the results corresponding to the *Community* and *Commercial* focus areas (Table 4). The organizations that scored less on these areas stated that these areas are of less importance for them, because the APIs are either targeted at internal users or the software platform itself is not yet mature enough. The area *Performance* also shows a wide range of results, indicating that the assessed platforms operate in different contexts. Such information could be preemptively collected through a checklist or survey, based on which the contents of the API-m-FAMM may be adapted. Other information that could be used to perform this adaptation may include characteristics such as the size of the organization, whether a third-party API management platform is used, or what type of product or services the organization provides.

The aforementioned opportunity for customization and adaptation of FAMMs can be linked to a general limitation of maturity models, that has been identified in as part of previously conducted research. In their work, Proença and Borbinha (2016) argue that *"maturity assessments can be used to measure the current maturity level of a certain aspect of an organization in a meaningful way, enabling stakeholders to clearly identify strengths and improvement points, and accordingly prioritize what to do in order to reach higher maturity levels"*. However, as a prerequisite to performing such a a maturity assessment, evidence needs to be manually collected to substantiate the maturity level calculation, which makes maturity assessment an expensive and burdensome activity for organizations to perform. Hence, Proença and Borbinha (2016) argue, future research should focus on developing methods and techniques to automate maturity assessment. The researchers second this observation: even though the maturity assessments that were done as part of the case studies were performed with minimal intervention of the researcher, there are opportunities for automation, customization, and adaptation of maturity models and FAMMs to reduce the amount of effort needed to perform maturity assessments and provide tailor-made advice for maturity improvement. Moreover, the researchers hypothesize that the aforementioned opportunities for automation, customization, and adaptation could be key in creating incentives for practitioners to re-use maturity models and FAMMs throughout a longer period of time. We consider maturity model's low degree of re-usability to be a point of concern due to our findings as part of the case study, which has shown that practitioners are relatively unlikely to revisit the API-m-FAMM to track their progress over a longer period of time.

The researchers are of the opinion that useful insights could be gained by conducting research into the differences in terms of advantages and disadvantages that occur with regards to API management for organizations that actively utilize third-party, commercial platforms to manage their APIs when compared to those who do not. This suspicion

is supported by the observation that currently, the largest part of available (grey) literature on the topic of API management is either written by authors that are either directly or indirectly affiliated to commercial management platform providers. Examples of such authors include Weir (2019) (director at Oracle) and De (2017) (former Apigee consultant). This literature is, more often than not, exclusively focused on the benefits that organization attain as a result of using API management platforms. However, when asked, some experts that were interviewed during the evaluation cycles noted that their organization does not use a management platform and does not wish to do so. For future work, it should be investigated whether significant differences exists in terms of API management maturity between organizations that do use commercial platforms and those that do not.

7. Focus Area Maturity Models

Throughout this work, the design, population, evaluation, and deployment of a focus area maturity model targeted towards the topic of API management has been described. Aside from the main practical contributions the API-m-FAMM offers organizations in maturing their API management practices, this work also provides various scientific contributions in the field of focus area maturity models.

This work provides researchers that seek to develop a focus area maturity model for a different functional domain with an improved description of the existing framework on how to do so. Publications on the development of FAMMs such as that of Jansen (2020) and Spruit and Röling (2014) offer a high-level overview of this process, which adhere to the FAMM meta-model presented by van Steenbergen et al. (2013) and De Bruin et al. (2005)'s methodology for developing maturity models. The accompanying source data of this article discusses the intermediate versionsof the API-m-FAMM and the changes between them in detail Mathijssen et al. (2021). The steps that this methodology consists of were also followed in this work: conducting a SLR, population, evaluation through expert interviews, and deployment as part of case studies. The added details contribute towards alleviating concerns mentioned by Jansen (2020): *"Interestingly enough, while there is a rapid increase of publications of new maturity models, there is little literature that particularly discusses the development of maturity models"*. Furthermore, work conducted by Proença and Borbinha (2016) on the state of the art of maturity models has found that one of the main limitations to maturity models is that there is a general *"Lack of information regarding the maturity model development method"*. This article could form an exemplar for other works describing FAMMs.

Because the underlying thought process and specific design choices that were made in developing other maturity models and FAMMs in particular was also largely inaccessible and unknown to the researchers, some novel approaches were used and described throughout this work. A card sorting technique was used and described, which uses digital tools to rank and assign practices to maturity levels. Additionally, the way these exercises were interpreted and the manner in which practices were ultimately assigned to maturity levels is elaborated upon. Another novel approach includes the utilization of criteria used for Design Sience Research (DSR) artefact evaluation introduced by Prat et al. (2015) to evaluate the API-m-FAMM's usefulness, completeness, ease of use, effectiveness, and operational feasibility. Doing so has shown that using these criteria is an adequate strategy for evaluating FAMMs during expert interviews or through surveys. In this work, the criteria were used to gather feedback and foster discussion with experts as well as among the researchers themselves, in order to subsequently guide further improvements to the API-m-FAMM. While in this case the criteria evaluation was conducted during the first evaluation cycle and as part of the case studies, the criteria could also be used to evaluate a prototype version of a FAMM to gauge interest among practitioners, or as part of the second evaluation cycle.

This work has shown that De Bruin et al. (2005)'s methodology for maturity model development may be incorporated with DSR, by using it to construct a design science artefact during the *solution design* phase. Furthermore, this study demonstrates which techniques and tools may be utilized during the *test* phase of De Bruin et al. (2005)'s methodology. This includes some of the approaches listed above, such as usage of Prat et al. (2015)'s evaluation criteria, conducting multiple evaluation cycles through expert interviews to evaluate the maturity model, as well as the usage of Nielsen (1995)'s card sorting technique to perform maturity level assignments. As such, this work provides researchers that utilize De Bruin et al. (2005)'s methodology and are involved with maturity model development with suggestions for the usage of novel approaches so that they may incorporate them in the testing phase of their maturity models.

Furthermore, the API-m-FAMM was successfully deployed in practice with minimal involvement of the researchers. As is described in Section 5, practitioners were provided with the API-m-FAMM, along with a set of instructions and a spreadsheet that was used to denote which practices had been implemented in the case company[3]. To the best of the researchers knowledge, this study is the first among work that has previously been done with regards to the design of FAMMs where practitioners are enabled to self-assess their organization's maturity in a functional domain such as API management. In comparison, Jansen (2020)'s SEG- M2 and Spruit and Röling (2014)'s ISFAM were deployed in practice by gathering input through in-person collaboration between the practitioners and the researchers themselves, as well as desk studies. Moreover, as is discussed in Section 5 and shown in Table 5, the majority of practitioners' experience with using the API-m-FAMM was positive, considering that its *ease of use*, *usefulness*, and *effectiveness* was ranked with a score of 4 out of 5 on average.

[3]These instructions and spreadsheet are available through the *do-it-yourself* kit, published on https://www.movereem.nl/api-m-famm.html.

8. Threats to Validity

Like all empirical research, this work is vulnerable to threats to validity. In this section we discuss four categories of validity: *construct validity*, *internal validity*, *external validity*, and *reliability* (Ampatzoglou et al., 2019; Runeson and Höst, 2009).

Construct validity refers to the degree to which this study actually measures what is is intended to. The API-m-FAMM sets out to evaluate the maturity of organizations with respect to API management practices. In order for this model to be successful and actually measure what we intended to, we need to trust in the application of focus area maturity models in general and the content of the API-m-FAMM specifically. This threat was mitigated through a number of actions. First, the SLR that was conducted as a means of initially populating the first versions of the API-m-FAMM adhered to several constraints. This includes a search string that was constructed through snowballing in an iterative manner. Furthermore, multiple databases were searched and strict inclusion and exclusion criteria were adhered to as to mitigate study inclusion and exclusion bias. Publication bias was further mitigated as a result of the inclusion of grey literature. Additionally, multiple discussion sessions were held among the authors at various stages of the population process in order to mitigate data extraction bias and researcher bias (Ampatzoglou et al., 2019). Moreover, construct validity is mitigated through this work's adherance to design science research (DSR) guidelines and De Bruin et al. (2005)'s methodology for developing maturity models. Lastly, robustness of the initial classification that was used by De (2017) was ensured by evaluating this decision through multiple expert interviews. Through these actions the API-m-FAMM builds on existing and proven research from SLRs, expert interviews, and maturity models.

Internal validity is concerned with the extent to which there is evidence that the artefact makes a difference in terms of cause and effect in the context of this study. The API-m-FAMM is intended to achieve both a realistic maturity evaluation as well as an actionable path for improvement. In order to investigate whether the model is able to achieve its intended goal, which is to assist organizations in assessing and evaluating their degree of maturity in API management in order to improve upon their API management related processes, the researchers suspect that multiple desk studies that last substantial periods of time (multiple years) should be conducted. However, verifying the implementation of practices contained in the API-m-FAMM through desk studies is largely infeasible since practices are often placed on long-term roadmaps that are susceptible to change, reducing the chance of actually observing the implementation of such practices during the desk study. Instead, the effects of the API-m-FAMM were investigated through evaluation during the experts interviews (Table 3) and case studies (Table 5). Considering the difference and increase in terms of the scores that were assigned to these criteria by practitioners when comparing the first and second version of the API-m-FAMM, in addition to the positive results and feedback received as a result of the case studies, we believe that the model is able to achieve its intended goal. In addition to this long-term goal, short-term benefits may also be attained by using the API-m-FAMM due to practitioners being able to immediately identify practices that are not currently implemented.

External validity revolves around the degree to which the results of this study may be generalized and applied to other contexts and situations. We conducted five case studies at four different companies. Based on these studies it is hard to claim that the API-m-FAMM will add value to all companies that expose public APIs. Although we are inclined to believe that our API-m-FAMM is an effective tool, based on the results presented, more studies need to be conducted over a longer time period. We do believe that the research design enables us to deploy the API-m-FAMM in a wide range of organizations. The API-m-FAMM is mainly targeted towards organizations that expose one or multiple public or partner APIs. However, two experts that are employed at organizations that currently exclusively utilize a set of internal APIs were involved in the first evaluation cycle. Considering that these experts expressed that focus areas such as *Security*, *Observability*, *Performance* and, to a lesser extent, some capabilities belonging to the *Lifecycle* focus area, are relevant and applicable to them, we suspect that the API-m-FAMM is also useful to such organizations. This is further supported by the results of the case study conducted at *ConsultComp*. Organizations of varying sizes and backgrounds were involved in the evaluation cycles and case studies. While some experts that are employed at large organizations noted that most practices have already been implemented at their organization, they also commented that some practices that are assigned to high maturity levels have not yet been implemented. Furthermore, experts working for small organizations indicated that the API-m-FAMM can be a useful tool for them to use as a road map or checklist to use in discussions with management and stakeholders so that future implementation of practices that are currently not implemented may be discussed and planned. Additionally, due to the decision to exclude practices that are solely tied to the usage of API management platforms, the API-m-FAMM is generalizable to both organizations that do not use such platforms, and those that do not. However, while this was not the case with organizations involved in the evaluation cycles and case studies, more case studies should be conducted with organizations that heavily utilize API management platforms to fully determine whether this exclusion in terms of scoping has negatively impacted the usability of the API-m-FAMM for such such organizations.

The **reliability** aspect is concerned with the extent to which the data and analyses that were conducted as part of these study are dependent on the specific researchers. A large part of the construction of the API-m-FAMM is the result of discussions among the authors. After every phase, as visualized in Figure 2, we integrated the new knowledge into the API-m-FAMM through discussions supported by the card sorting technique. However, we designed the research to be as transparent as possible, and published all intermediate versions and source data. Due to adherence to DSR guidelines,

the use of De Bruin et al. (2005)'s methodology, and the SLR, some degree of researcher bias has been mitigated. Furthermore, the protocols that were used for experts interviews and case studies are included and were reviewed through peer reviewing among all involved researchers. Additionally, all design decisions and processes that resulted in the various increments of the API-m-FAMM are extensively documented, along with separate source documents detailing each major version of the model.

Another threat to reliability is that, considering the maturity level assignments described in Section 3 were partially done in a pragmatic and subjective matter, they may result in a different outcome if replicated. This is due to the fact that the experts that ranked practices as part of the maturity ranking exercises each have varying backgrounds, experiences, and are employed at organizations with different characteristics. However, this threat was mitigated by using the average of these maturity assessments, as well as analyzing and cross-evaluating focus areas and maturity assignments during the second evaluation cycle. Furthermore, the deployment of the API-m-FAMM as part of the case studies has not produced any criticism among practitioners on the maturity levels that practices are assigned to.

9. Conclusion

Throughout this work, the design, population, evaluation, and deployment of a focus area maturity model targeted towards the topic of API management has been described. The goal of this model as well as this work in general was to improve the transparency and availability of API management assessment frameworks and tools by constructing, evaluating and validating a publicly available, industry and academically grounded framework or tool that can be used by organizations that expose their API(s) to third-party developers to assess and evaluate their degree of maturity with regards to API management in order to improve upon their API management-related business processes. By constructing, evaluating, and publishing the API-m-FAMM we answered the research question posed in Section 3: *How can organizations that expose their APIs to third parties evaluate their API management practices?*. Aside from the main practical contributions the API-m-FAMM offers organizations in maturing their API management practices, this work provides the following scientific contributions.

Firstly, this work offers researchers a previously undefined framework that captures the topics and processes API management consists of. By decoupling API management processes and topics from commercial platforms, the API-m-FAMM offers the academic community a de-commercialized overview of the topic that was developed by using insights from both literature as well as the industry. Secondly, this work provides a detailed description of the construction of a focus area maturity model through the published source data (Mathijssen et al., 2021), that researchers can use as an example in future works. Thirdly, this work has shown that De Bruin et al. (2005)'s methodology for maturity model development may be incorporated with DSR, utilizing Nielsen (1995)'s card sorting technique to perform maturity level assignments, conducting multiple evaluation cycles, and utilization of criteria used for DSR artefact evaluation introduced by Prat et al. (2015). Lastly, The API-m-FAMM was successfully deployed in practice with minimal involvement of the researchers using the constructed *do-it-yourself* kit. This shows that we as researchers can make maturity models more relevant for industry by investing in the usability of these assessment and improvement tools.

References

Ampatzoglou, A., Bibi, S., Avgeriou, P., Verbeek, M., Chatzigeorgiou, A., 2019. Identifying, categorizing and mitigating threats to validity in software engineering secondary studies. Information and Software Technology 106, 201–230.

Arsanjani, A., Holley, K., 2006. The service integration maturity model: Achieving flexibility in the transformation to soa, in: 2006 IEEE International Conference on Services Computing (SCC'06), IEEE. pp. 515–515.

Basole, R.C., 2019. On the evolution of service ecosystems: A study of the emerging api economy, in: Handbook of Service Science, Volume II. Springer, pp. 479–495.

Becker, J., Knackstedt, R., Pöppelbuß, J., 2009. Developing Maturity Models for IT Management. Business & Information Systems Engineering 1, 213–222.

Bui, D.H., 2018. Design and Evaluation of a Collaborative Approach for API Lifecycle Management. Master's thesis.

CA Technologies, 2019. The API Management Playbook: Understanding Solutions for API Management. URL: https://docs.broadcom.com/docs/the-api-management-playbook.

Carmel, E., Agarwal, R., 2006. The maturation of offshore sourcing of information technology work, in: Information systems outsourcing. Springer, pp. 631–650.

Coleman Parkes Research, 2017. Apis: Building a connected business in the app economy. URL: https://docs.broadcom.com/doc/apis-building-a-connected-business-in-the-app-economy.

Crawford, J.K., et al., 2007. Project Management Maturity Model. Taylor & Francis.

De, B., 2017. API Management. Springer.

De Bruin, T., Rosemann, M., Freeze, R., Kaulkarni, U., 2005. Understanding the main phases of developing a maturity assessment model, in: Australasian Conference on Information Systems (ACIS):, Australasian Chapter of the Association for Information Systems. pp. 8–19.

Devoteam, 2016. API Management at Liberty Global Inc. URL: https://nl.devoteam.com/en/our-case-studies/api-management-liberty-global-inc.

Endjin, 2017. API Maturity Matrix. URL: https://endjin.com/blog/2017/07/kickstart-your-api-proposition-with-the-api-maturity-matrix.

Etikan, I., Musa, S.A., Alkassim, R.S., 2016. Comparison of convenience sampling and purposive sampling. American journal of theoretical and applied statistics 5, 1–4.

Gámez Díaz, A., Fernández Montes, P., Ruiz Cortés, A., 2015. Towards SLA-driven API Gateways. XI Jornadas De Ciencia E IngenierÍA De Servicios .

Gartner, 2019. A Guidance Framework for Evaluating API Management Solutions. URL: https://www.gartner.com/en/documents/3956412/a-guidance-framework-for-evaluating-api-management-solut.

Haddad, C., 2015. Comparison Evaluation Matrix. URL: https://wso2.com/whitepapers/comparison-evaluation-matrix/.

Hüner, K.M., Ofner, M., Otto, B., 2009. Towards a maturity model for corporate data quality management, in: Proceedings of the 2009 ACM symposium on Applied Computing, pp. 231–238.

Iversen, J., Nielsen, P.A., Norbjerg, J., 1999. Situated assessment of problems in software development. ACM SIGMIS Database: the Database for Advances in Information Systems 30, 66–81.

Jansen, S., 2020. A Focus Area Maturity Model for Software Ecosystem Governance. Information and Software Technology 118, 106219.

Jansen, S., Brinkkemper, S., Cusumano, M.A., 2013. Software ecosystems: Analyzing and managing business networks in the software industry. Edward Elgar Publishing. doi:10.4337/9781781955635.

Kitchenham, B., Charters, S., 2007. Guidelines for performing systematic literature reviews in software engineering. Keele University and Durham University Joint Report .

Lees, C., Mallick, A., Paar, T., Fontenay, E., Pimakova, K., 2019. APIs: The Digital Glue - How Banks Can Thrive in an API Economy. URL: https://www.accenture.com/_acnmedia/PDF-100/Accenture-How-Banks-Can-Thrive-API-Economy.pdf.

Mathijssen, M., Overeem, M., Jansen, S., 2020. Identification of practices and capabilities in api management: A systematic literature review. arXiv preprint arXiv:2006.10481 .

Mathijssen, M., Overeem, M., Jansen, S., 2021. Source data for the focus area maturity model for api management. arXiv:2007.10611.

Medjaoui, M., Wilde, E., Mitra, R., Amundsen, M., 2018. Continuous API Management: Making the Right Decisions in an Evolving Landscape. O'Reilly Media.

Mettler, T., Rohner, P., Winter, R., 2010. Towards a classification of maturity models in information systems, in: Management of the interconnected world. Springer, pp. 333–340.

Nielsen, J., 1995. Card sorting to discover the users' model of the information space. Nielsen Norman Group .

Okoli, C., 2015. A Guide to Conducting a Standalone Systematic Literature Review .

Paulk, M.C., Curtis, B., Chrissis, M.B., Weber, C.V., 1993. Capability maturity model, version 1.1. IEEE software 10, 18–27.

Poell, T., Nieborg, D., van Dijck, J., 2019. Platformisation. Internet Policy Review 8, 1–13. doi:10.14763/2019.4.1425.

Pöppelbuß, J., Röglinger, M., 2011. What makes a useful maturity model? a framework of general design principles for maturity models and its demonstration in business process management .

Prat, N., Comyn-Wattiau, I., Akoka, J., 2015. A taxonomy of evaluation methods for information systems artifacts. Journal of Management Information Systems 32, 229–267.

Proença, D., Borbinha, J., 2016. Maturity models for information systems: A state of the art. Procedia Computer Science 100, 1042–1049.

Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empirical software engineering 14, 131.

Santos, W., 2019. Apis show faster growth rate in 2019 than previous years. URL: https://www.programmableweb.com/news/apis-show-faster-growth-rate-2019-previous-years/research/2019/07/17.

Spruit, M., Röling, M., 2014. ISFAM: The Information Security Focus Area Maturity Model .

van Steenbergen, M., Bos, R., Brinkkemper, S., van de Weerd, I., Bekkers, W., 2013. Improving IS Functions Step by Step: the Use of Focus Area Maturity Models. Scandinavian J. Inf. Systems 25, 2.

Tung, T., 2014. Accenture API Management Suite API Maturity Model. URL: https://expydoc.com/doc/4730985/download-pdf.

Van Steenbergen, M., Bos, R., Brinkkemper, S., Van De Weerd, I., Bekkers, W., 2010. The Design of Focus Area Maturity Models, in: International Conference on Design Science Research in Information Systems, Springer. pp. 317–332.

Weir, L., 2019. Enterprise API Management: Design and Deliver Valuable Business APIs. Packt Publishing Ltd.

WSO2, 2015. API Management Platform Technical Evaluation Framework. URL: https://wso2.com/whitepapers/api-management-platform-technical-evaluation-framework/.