A Focus Area Maturity Model for Software Ecosystem Governance

Slinger Jansen

Slinger.Jansen@uu.nl, Utrecht University, Princetonplein 5, 3584CH Utrecht

Abstract

Context. Increasingly, software companies are realizing that they can no longer compete through product excellence alone. The ecosystems that surround platforms, such as operating systems, enterprise applications, and even social networks are undeniably responsible for a large part of a platform's success. With this realization, software producing organizations need to devise tools and strategies to improve their ecosystems and reinvent tools that others have invented many times before.
Objective. In this article, the software ecosystem governance maturity model (SEG-M^2) is presented, which has been designed along the principles of a focus area maturity model. The SEG-M^2 has been designed for software producing organizations to assess their ecosystem governance practices, set a goal for improvement, and execute an improvement plan.
Method. The model has been created following an established focus area maturity model design method. The model has been evaluated in six evaluating case studies with practitioners, first by applying the model to their organizations and secondly by evaluating with the practitioners whether the evaluation and improvement advice from the model is valid, useful, and effective.
Result. The model is extensively described and illustrated using six desk studies and six case studies.
Conclusions. The model is evaluated by both researchers and practitioners as a useful collection of practices that enable decision making about software ecosystem governance. We find that maturity models are an effective tool in disseminating a large collection of knowledge, but that research and creation tooling for maturity models is limited.

Keywords: Software ecosystem governance, developer ecosystems, focus area maturity models

1. Introduction

The concept of a software ecosystem has made a large impact on the platform business and research world. Over a short period of time both scientists (Bosch et al. [1], Manikas et al. [2], Alves et al. [3], Mens et al. [4], and many others) and companies (Apple, Microsoft, SAP, etc.) have conceptualized and realized software ecosystems in a way that has significantly affected society and the software industry. Software producing organizations are structurally trying to improve their position in their software ecosystems. After all, having a top selling app in an app store can mean instant success for budding software companies and being the number one platform in a particular domain provides longevity and propensity for growth [5].

1.1. Motivations for Ecosystem Governance

Software ecosystems are an effective way to construct large software systems on top of a software platform by composing components developed by actors both internal and external [6]. Olsson and Bosch [7] define the following motivators for developing software ecosystems. We elaborate their list of motivators with findings from our previous work [8].

First, ***customers can demand diversity and variety*** in a large-scale product, where the organization supporting the product wants to focus on the core features of the product. They also observe that the ***costs of innovation can be shared*** and spread throughout the value chain. We find that that is true for three reasons. (1) Extenders can combine technologies that enable innovations that the platform orchestrator would not have implemented, such as a hardware extension for a mobile phone that scans bar codes. (2) Extenders can focus on niches that the platform orchestrator cannot focus on, such as a paper invoice scanning software company, that builds an extension for a bookkeeping platform. (3) When companies specialize, they inevitably become more successful in their niche than a generalist, which means that the platform provider can focus on their specialty: building a platform.

Secondly, Olsson and Bosch suggest that the costs for commoditizing functionality can be decreased by ***sharing maintenance***. Apple, for instance, did not want to be a game studio, Enterprise Resource Planning vendor, and media company, besides developing high quality phones and an operating system. The niche players, on the other hand, did not want to have to develop new gaming devices

that inevitably would have been less successful than Apple's platforms. These extenders are well willing to pay the 30% markup on their extensions in the app store [9]. On the other hand, the platform provider can ***extract value from the ecosystem***.

Thirdly, Olsson and Bosch observe that extender technologies may also become platformized in the long run. This ***creates ecosystems of ecosystems***, where new platforms are formed on top of existing platforms. An interesting example is the ARComp case that is presented in this article, which is an augmented reality platform based on the Unity platform.

We add two further observations. Organizations can use reseller ecosystems for economies of scale and to ***penetrate hard to reach markets***. Microsoft, for instance, in its early years realized that they could capture markets faster using partners. Also, SAP always wanted to be a platform company, and not a services company, and thereby developed one of the largest partner networks in the world, servicing SAP customers worldwide.

Another motivation for organizations to invest into management of a software ecosystem is ***Staying Power*** [5]: customers who have invested significantly in the connection of their systems will be less likely to leave the ecosystem. Iansiti and Levien [10] mention that "if a SECO orchestrator continually improves their platform SECO, they ensure their own survival and prosperity." An extension of that is when partners invest in the platform and attract new sales to the platform. Intel, for instance, has created an investment fund of more than one billion euros to make sure young high potential companies use Intel technologies for their future best sellers to attract these budding companies to their ecosystems for years to come. Besides an increased change of business survival, a SECO is a powerful source of competitive advantage for an orchestrator. According to Williamson and De Meyer [11], an orchestrator may reap the benefits of economies of scale by creating a platform ecosystem. This requires a lower investment than if the orchestrator would try to offer the functionality itself.

1.2. Definitions

Several terms need to be defined. We define a software ecosystem as a set of organizations collaboratively serving a market for software and services [8]. Typically these ecosystems are underpinned by a common technology, such as an extendable software platform. Furthermore, an assumption is made that there exists an ecosystem orchestrator, that develops a platform and orchestrates the ecosystem around it. The orchestrator is also often called a keystone in an ecosystem. Partners are organizations that provide services and products that extend the capabilities of the orchestrator, such as consultancy, product, or knowledge partners. Extenders are organizations that extend a platform with an extension: an application or solution that further builds on the platform. Extenders are not always acknowledged partners by the orchestrator, such as single open source developers or competitors developing extensions. Furthermore, the term app is used interchangeably with application, solution, or extension.

Alves et al. [12] define software ecosystem governance mechanisms as managerial tools of players in software ecosystems that have the goal of influencing an ecosystem's health. Furthermore, they categorize governance practices at a course grained level into the practices of "Value Creation", "Coordination of Players", and "Organizational Openness and Control". In this article, the governance definition of Alves et al. is followed and the governance tools that are expressed in their work form an inspiration to the model presented in this article.

1.3. Problem Statement and Contributions

Even though the field of software ecosystem management and software ecosystem governance is rapidly maturing, many organizations are still reinventing tools and methods for becoming stronger in a software ecosystem. There exists little usable knowledge on quickly implementable processes and practices for organizing an ecosystem. The organizations that participated in this study indicated that they lacked a comprehensive framework for the tools and practices available to them to improve and advance the management of their software ecosystems. This leads to the following research question: *"How can a maturity model be developed that enables organizations to assess and advance their software ecosystem governance practices?"*

This work provides three main contributions.

- In Section 2 the research method is presented. The research method section discusses the effectiveness of a maturity model as a vehicle of disseminating knowledge about software ecosystem governance. One of its main contributions is that the section discusses **how the maturity model is populated with literature and evaluated through desk and empirical case studies**.
- Section 4 **presents the six empirical case studies at four companies** and highlights their motivations for developing large ecosystem governance improvement initiatives. Furthermore, a cross-case synthesis provides insight into the practices organizations implement most frequently and what types of organizations are mature in their software ecosystem governance.
- In Section 3 **we present the Software Ecosystem Governance Maturity Model (SEG-M^2)**. The maturity model provides concrete and detailed instructions on how to assess a company's maturity in terms of governance of its ecosystem. Furthermore, it provides concrete instructions on how to move the software ecosystem governance initiatives forward. It provides organizations with concrete tools, processes, and methods for creating and developing a software

ecosystem. Furthermore, it discusses which maturity levels can be achieved and who are the roles responsible for the process.

- **We provide detailed descriptions of the SEG-M^2 practices** [13] that organizations can implement, including their sources in literature. The detailed descriptions have the goal to disambiguate the practices and provide organizations with handles to self-evaluate their software ecosystem governance practices.

In Sections 6 and 7, we discuss and summarize our contributions. The main points of validity are that maturity models are an appropriate vehicle for communicating extensive domain knowledge and that there exists insufficient tooling for maturity models. We also hypothesize about the patterns observed in more mature software producing organizations and that open and closed software platforms are not significantly different when it comes to software ecosystem governance. Furthermore, we highlight how an organization's strategy determines the target maturity level for any software ecosystem governing organization.

2. Research Method

The SEG-M^2 has been created following the guidelines for creating focus area maturity models [14]. We have chosen focus area maturity models [15] as the framework for structuring our findings. Focus area maturity models are especially effective at defining a domain and providing organizations with sets of implementable practices and processes. Focus area maturity models are distinguished from fixed-level maturity models, such as the capability maturity model for software development, in that they are especially suited to the incremental improvement of functional domains. To be more specific, the SEG-M^2 has been created following the guidelines for creating maturity models, as presented by de Bruin et al. [14]. The SEG-M^2 went through two evaluation cycles. In the first cycle, we evaluated the cases against sixdesk studies, which looked at existing materials of existing companies, mostly by literature study, old case materials, and online platform descriptions. In the second cycle, the SEG-M^2 was evaluated and complemented using empirical case studies, each comprising 5 days or more on site, through six software ecosystem governance maturity evaluations at four companies. The model was not significantly changed after the first cycle. Saturation was not purposefully reached, but the case participants all indicated that the model was useful to them.

The work is the summarization and culmination of our previous work in this domain. In the past we frequently discuss the governance and management of software ecosystems from the keystone point of view. In 2010 van den Berk and Jansen [16] created a model to assess the strategy of companies in software ecosystems, using metric categories of biology, lifestyle, environment, and health care organization. van den Berk's model is course grained and hard to adopt in practice, especially because few actual governance mechanisms are provided. In our work on defining ecosystems [8] a model is provided that also includes governance mechanisms for both open source and closed organizations. The practices in that framework have also been foundational to this work, although the governance practices remain abstract, e.g., "Form alliances", "Create a partnership model", and "Start certification program".

Secondly, the work of Baars and Jansen [17] comes close to the SEG-M^2 that is presented in this article. Baars' work was inspired in part by our previous work on defining software ecosystems [8]. It is the first model of its kind that presents concrete governance practices for keystone firms, such as "Create a development standard", and is evaluated in two case studies. Many of the practices from Baars et al. have been adopted in the SEG-M^2, although the practices in this work are more fine-grained and more information is provided how the practices need to be implemented. Furthermore, the work of van Angeren et al. [18] attempts to compare four similar ecosystems from two large software vendors and find the differences in governance mechanisms. These works have all been fundamental in the identification of governance practices in software ecosystems as identified in this work.

2.1. Focus Area Maturity Models

Maturity models are a proven tool in the creation of collections of knowledge of practices and processes about a particular domain [19]. Examples of maturity models are the project management maturity model [20], the capability maturity model for software development [21], the Industry Open Source Model [22], and the service integration maturity model [23].

One specific type of maturity model, focus area maturity models [15, 24], is used to establish the maturity levels of an organization in a specific functional domain. A focus area maturity model must have a well-defined scope in the sense of the functional domain it applies to. A functional domain is described by the set of focus areas that constitute it. With each focus area a set of capabilities is associated. The capabilities are positioned against each other in a maturity matrix. Based on the positioning of the capabilities in the maturity matrix a number of maturity levels can be distinguished. To guide the organization in incremental development of the functional domain, improvement actions are associated with the capabilities. A simplified meta-model for maturity models is given in figure 1, highlighting the main concepts of maturity models: *focus areas, capabilities, practices*, and *maturity levels*. The focus area maturity model for software ecosystem governance has seven focus areas, 38 capabilities, 168 practices, and eight maturity levels for each of the focus areas (including level 0).

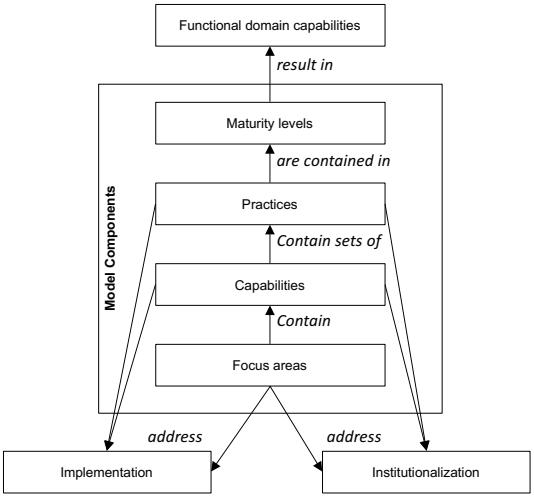

Figure 1: A meta-model for focus area maturity models.

de Bruin et al. [14] generalize a method for creating a maturity model from existing models. They propose the phases of *scope, design, populate, test, deploy, and maintain*. We follow their process to describe the creation of the SEG-M^2.

- **Scope -** The focus of the SEG-M^2 is domain specific: it aims to describe governance practices of ecosystem coordinators, i.e., people who are responsible for the ecosystem; its design, its management, and its performance. The audience consists of practitioners, although academics may find inspiration from the SEG-M^2 to research particular practices and their effects on the health of ecosystems.
- **Design -** The design phase attempts to answer the why, how, and who questions.
  - **The 'why'** for the SEG-M^2 is that its main goal is to support organizations that own a platform, in maturing their ecosystem governance practices.
  - **The 'how'**, is that organizations can implement particular practices to reach a particular level of maturity in a focus area.
  - **The 'who'** is actually a two-way question: to whom does the SEG-M^2 apply and who applies the SEG-M^2. The audience is both managers of partner management programs, for instance sales, consultancy, or development, and managers of technical departments of organizations, such as CTOs, development program managers, and community managers. In this article we present how the SEG-M^2 is applied to evaluate different cases. The model has been designed with the purpose to be applied in self-assessments by organizations, as well. Two other researchers have applied the SEG-M^2 and reported that it is doable with the detailed practice descriptions [13].
- **Populate -** The practices were found by taking the literature studies of Manikas [2] and Alves et al. [12] as a starting point and snowballing forward and backward [25]. Please note that these also include our previous studies on software ecosystem management and governance [17, 18, 8, 16]. We analyzed the papers mentioned in these studies and identified the practices in them. Subsequently, we snowballed one level deeper. To supplement the study with more recent articles, we also added the articles that cited these two literature studies to our literature body.

  We defined a practice as *any practice that has the express goal to change the position of the platform in the software ecosystem*, for instance by mobilizing and attracting more developers. Furthermore, the practice has to be executable by a member of the platform team and should have a defined owner. The practices were positioned into the maturity model pragmatically. Some of the practices would have fit at different levels and the design decision was made to put practices in a separate box, i.e., no box is filled by multiple practices. When required, an extra focus area was introduced. Dependencies were avoided where possible, although they are present in the SEG-M^2. There is a 'natural' progression of the practices, and oftentimes a level 4 practice cannot be introduced without having first implemented a lower level practice. Some of the practices were moved to different levels during the evaluations. For example, with the availability of particular tools, such as documentation generation tools from API specifications, it became significantly easier to create interactive documentation, we found. That practice was moved from level 7 to level 4 during the evaluations.
- **Test -** The SEG-M^2 has been evaluated in two rounds. First, six desk studies were performed. These cases were selected by the organizations that participated in the evaluations, i.e., they wished to be benchmarked against these particular organizations (Microsoft, Salesforce.com, Eclipse, iOS, Android, Cisco). The practices were evaluated using documentation, creating developer accounts, and informal interviews with developers within these ecosystems. These reference cases provided benchmarking capabilities to the SEG-M^2. six empirical cases at four companies followed the case study protocol described below.
- **Deploy -** The deployment of the SEG-M^2 follows two steps. First, the evaluation at the case companies has functioned in part as an extra evaluation step, but also to test the adoption and acceptance of the SEG-M^2

in practice. The second part is the publication of the SEG-M^2, of which this article is the main artefact.

- **Maintain -** The model is currently in its first public version. In the next phase, the SEG-M^2 should be applied more often to gather evidence. Furthermore, a discussion platform is needed to encourage discourse about the SEG-M^2. We consider putting the practices and the SEG-M^2 into a curated Wiki as future work.

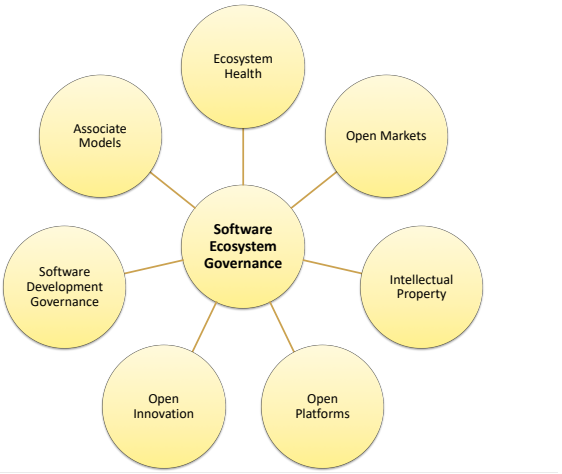

Figure 2: Seven focus areas of the SEG-M^2.

The maturity levels in table 1 have been defined as ambition goal levels. These levels have been determined pragmatically: they are goals that were defined by the organizations who participated in the case studies. There exists a relationship between the levels and the practices. In many cases, the practices that are necessary to reach a particular maturity level, and do not make sense for other levels. For instance, being a leading ecosystem requires that the architecture of the platform that is managed needs to be continuously hardened (*practice 5.1.4*). At lower levels that practice does not necessarily need to be implemented. That said, it is still challenging to associate higher levels with practices, as being "an ecosystem of ecosystems" is not something that is simply reached by implementing all the practices. One would still need to attract a significant number of developers and partners before that level is reached. Fortunately, evidence shows that organizations only fully implement the practices when they are needed and fit a particular need in the ecosystem. Please note that organizations do not achieve one level during an evaluation: they achieve a level for each focus area. Table 1 is best interpreted as a table of ambition levels and if an organization is not interested in a particular focus area, they do not need to achieve the target level in that focus area.

Please note that some of the case company teams indicated that they do not wish for their commercially sensitive information to be published and have requested anonymity. For some of the case companies, however, it is easy to retrieve their identity. ARcompP1, for instance, has already been mentioned as Vuforia in another publication. We have uniformly named the empirical case companies with code names, such as NetCompP1. The desk studies have been identified by name, as the material used for these cases is publicly available.

2.2. Test Phase and Validity: Case Study Approach

Before publishing the SEG-M^2, it was tested on six case studies at four companies. The case studies followed the steps defined by Yin [26]. The case studies were found using convenience sampling; organizations approached members of our research group to establish whether we could support them in the improvement of their software ecosystem governance practices.

The case studies lasted five days or more per assessment. The organizations that participated in this work had a strategic focus on their software ecosystems. Because of top management buy-in, it was not hard to convince the companies to participate in these studies.

On the first day, the researcher studied the ecosystem independently, doing research into the business model (using Schief's software business model framework [27]), the market situation, the platform orchestrator, and by evaluating competing platforms. Assumptions and observations were noted down in the case report, to be confirmed during the later interviews. Typically, the case study was launched in a group meeting, where the platform's main managers were present. Such groups typically consisted of a director, one or more product managers, a release manager, a quality assurance manager, a community manager, an account manager, and in many cases a market manager, e.g., the manager of the app store. Over the following days, new interviewees were identified in a snowballing manner during the interviews, or by the managers of the platform.

Over the course of three days interviews were organized with platform developers, market developers, quality assurance team members, and the managers mentioned above. Interviews lasted between one and three hours. Longer interviews typically also involved translators at companies where the local language was not English. Due to anonimization we cannot disclose which languages these were.

The interviews followed a protocol consisting of three parts: first the goal of the research was presented. Secondly, the position of the interviewee was discussed. Thirdly, each of the SEG-M^2 practices was discussed. Once a practice had been identified to be present or absent without conflict twice in interviews or documentation, the practice was not further discussed in following interviews, to ensure that enough time was left to discuss other practices. Interviewees commented on the individual prac-

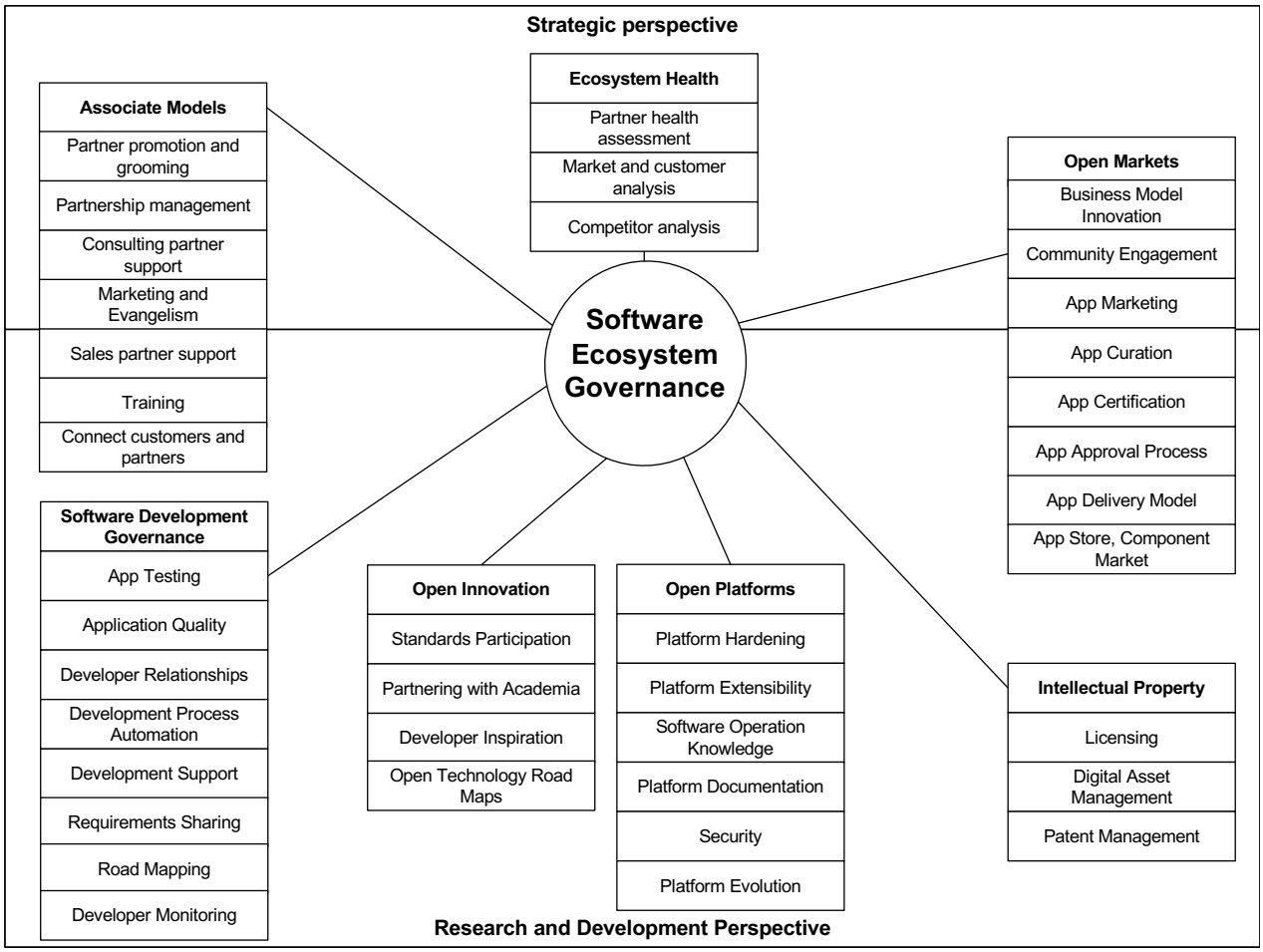

Figure 3: the SEG-M^2, its key processes, and its focus areas. Please note that this is an expanded version of the model shown in Figure 2.

tices and were often interested in receiving more materials around the work that they did.

After processing the interviews over the course of several days, a workshop was organized with the members of the team, describing how the SEG-M^2 could be used to advance the company's ecosystem management. First, a presentation was given about the assessment, the role of the assessment, and the outcome. Next, over the course of four hours the practices were discussed that should be implemented to mature ecosystem management within the company. Practices were marked as 'implementable', 'implementable and planned', 'not applicable', and 'not implementable'.

The case studies were performed by three different researchers; the first author and two researchers active in the domain. The results of each of the case studies were reviewed by one other researcher, to evaluate each other's findings. Typically, very few changes were made after one of these reviews. The data of the cases is available, but due to the commercially sensitive nature, notes, interviews, and evaluations can only be reviewed under the supervision of the authors. The case material is organized in separate digital folders per case.

3. The Software Ecosystem Governance Maturity Model

The practices are divided into seven focus areas and are modeled in figure 2. The seven focus areas were identified by classifying the practices according to topic. The focus areas are not of equal size, as the topic of software ecosystem governance is not equally distributed over different domains; it for instance has many practices around software development and only some minor but relevant practices around intellectual property. Furthermore, the division between the *strategic perspective* and **research and development perspective** was validated by discussions had with the case participants. These participants typically fell into the categories of more technically oriented staff versus the more management oriented staff.

Table 1: Ambition maturity levels in the SEG-M^2 per focus area. Please note that the examples do not score this level on all of the focus areas; the examples serve as an illustration of the kinds of companies that achieve these maturity levels in most of the focus areas.

Level	Name	Description	Examples
0	**No ecosystem**	Products are budding open, first lists of partners are being created, but the ecosystems are unstructured and the coordinating organization is immature.	NetCompP3, NetCompP1
1	**Extensible open product**	Products are opened up for multi-layer extension, but very little attention is paid to ecosystem coordination. Partner management receives little attention.	NetCompP2, ERPCompP1
2	**Extensible open platform**	The product is increasingly seen as a platform. Third parties approach the organization with feature requests about the platform.	XBMC platform
3	**Robust platform ecosystem**	Partners increasingly base their business on the platform. The platform is leading in some niches. The supporting organization can fully support all partners. Some certification takes place.	Eclipse platform
4	**Leading ecosystem**	Partners are benefiting greatly from the platform. Customers are increasingly seeing the value of the platform and creating extensions themselves. The platform is challenging the status quo in some industries.	SAP Hana Platform
5	**Reigning ecosystem**	The ecosystem is at full strength and growing rapidly. Partners are experiencing strong connections with the coordinating firm. The coordinating party is strategically focusing on the platform and decreasing its efforts on customers and end-users. It is on top in many industries and seen as a market leader. Others say they want to have an ecosystem such as the coordinating party.	Steam platform
6	**Absorbing ecosystem**	The ecosystem is leading and also absorbing other ecosystems, such as surrounding hardware and software ecosystems. Patents and mergers have become strategic instruments for increasing business. New niches are introduced regularly.	Apple iOS Platform
7	**Ecosystem of ecosystems**	The ecosystem is absorbing other ecosystems and creating new ones in its wake. Third parties can create markets in markets. The coordinating firm needs to maintain an open strategy for fear of monopoly.	Google Android Platform, RedHat Linux platform

Fundamentally, software product and platform producing organizations all want the same thing: to run an innovative continuous software business with propensity for growth. Cusumano already dubbed this term "Staying Power" [5]. The construction of an ecosystem around a platform is perhaps the epitome of Staying Power. Staying Power is reached by minimizing risk, increasing innovation, increasing revenue, and creating a healthy network of partners around the business. Each of the seven focus areas in the SEG-M^2 is represented by these core values when improving Staying Power. The seven focus areas are, going clockwise in figure 2:

- **Associate Models -** All practices to do with management and coordination of partners is found under the associate models focus area. It contains practices such as the creation of partnership models, partner training, and consultancy and sales partner support. One of the more technical aspects of associate models is the creation of systems that enable partners to communicate with end users, such as approval systems in app stores or SAP's customer partner connection center, that enables partners to share ticketing systems with customers and SAP itself.

- **Ecosystem Health -** The ecosystem health perspective regards the ecosystem as a living ecosystem that can be analyzed as a whole, also contrasting itself with other potentially influencing ecosystems. The practices in this focus area are concerned with partner health analysis, sharing of market data, and making strategic choices in regards to competing ecosystems.

- **Open Markets -** The open Markets focus area concerns itself with the creation of an open market for services and applications. The practices belonging to extension approval, extension marketing, business model innovation, and app delivery are part of the open markets focus area. The area evenly divides itself across management and technical boundaries.

- **Open Platforms -** All practices related to the creation of a stable solid and open platform belong to the open platforms focus area. It is concerned with the creation of a platform, the platform's security, its extension capabilities, and documentation.

- **Intellectual Property -** The practices to do with patent management and intellectual property management within the ecosystem are gathered in the focus area around intellectual property. At the lowest levels it is concerned with innovation sharing across the ecosystem. At the higher levels it is concerned with patents, licenses, and stimulation of ecosystem health by co-creation.

- **Open Innovation -** The open innovation focus area is concerned with sharing knowledge across the

ecosystem to feed external developers with new possibilities for improvement, also known as niche creation. At the lowest levels it is concerned with sharing development practices and innovations with partners. At higher levels it is concerned with creating shared innovations and ecosystem standards.

- **Software Development Governance -** In this focus area, all practices are collected that are concerned with observing, supporting, and enabling software developers. The practices are concerned with domains such as testing, road mapping, shared requirements. At the lowest levels the focus area is concerned with opening up to developers and enabling them to develop third-party extensions. At higher levels it is concerned with collecting data (software operation knowledge, or SOK [28]) about applications and their developers and about supporting developers in helping each other.

Under the seven ecosystem management focus areas 168 practices have been identified. These practices have been collected into an ecosystem management maturity model, with the goal of providing ecosystem managers with a road to improvement and achievements of higher levels of ecosystem management maturity. Eight ecosystem management maturity levels have been established, as listed in table 1.

The practices that are in the SEG-M^2 are listed in Tables 3 and 4. Furthermore, they have been described in detail [13] as to provide both practitioners and researchers with a full background on the practice. The practices are deeply rooted in both empirical experience, the desk studies, and literature. The practices have been described using the following elements:

- **Practice code -** The practice code is made up of three numbers. The first number concerns the focus area, the second number the capability, and the third number the maturity level. As there are empty elements in the matrix, the numbers are not consecutive.
- **Practice -** The name of the practice, as it is mentioned in the SEG-M^2.
- **Focus area -** The focus area is mentioned to indicate the domain in which this practice is relevant.
- **Description -** A paragraph of text is provided to describe the practice in detail. The main reason for providing a lengthy description is internal validity: in future evaluations by third parties, they should be able to perform the evaluations independently.
- **When implemented -** Provides a series of necessary conditions before this practice can be marked as implemented. Again, to strengthen internal validity of the SEG-M^2.
- **Role responsible -** One of the main findings during the case studies was that managers wanted to know who should be responsible for implementing a particular practice. This is now part of the SEG-M^2 as well. The roles are indicators, as the naming in companies can be different and domain specific.
- **Literature -** Several references are given to articles that mention the practice. The literature is mainly found in the mentioned SLRs.

In Table 2 a sample of a practice description is provided.

3.1. Roles in Software Ecosystem Governance

There are several different roles that play a part in governing the ecosystem. Examples are product managers, quality managers, and release managers. These may traditionally have had an inward role, but now need to facilitate partners as well as customers. In the practice descriptions the roles are explicitly appointed: largely to make sure that there are people adopting practices as their own responsibility instead of leaving it to the group. These roles are given for inspiration, as not all of these may be available within a software producing organization.

- **Chief Technology Officer -** The chief technology officer (CTO) is an executive-level decision maker throughout the whole process of software ecosystem development [8]. The CTO concerns long-term and "big picture" issues focusing on the focal company and other supporting technologies involving participants within the ecosystem. In the meanwhile, the CTO may focus on commercialization of certain platform or technologies. Moreover, the CTO should also be in charge of the technical personnel (i.e., developers and technical support group) management and policy establishment.
- **Chief Software Architect -** The chief architect is one of the most important decision makers in the process of establishing a software ecosystem [31, 30, 32] . The chief architect determines how the platform will be extended by extenders and enables an open, extendable, innovative, smartly versioned architecture. The chief architect is involved in the process of API and SDK design, establishing where the best monitoring points are, and how the system must be made as secure as possible, while remaining open for extension.
- **Software Product Manager -** The software product manager or product owner is the first person who starts incorporating wishes from partners into a product or platform. The job of a software product manager is significantly changed when an ecosystem is introduced for a platform [8]. The tasks the software product manager will be executing are the creation of open requirements management systems, the creation of documentation for extenders to create extensions

Table 2: Example Practice Description: Share AppTest procedures

Practice code:	Name:	Focus area:
2.1.3	**Share AppTest procedures**	**Software Development Governance**
Description: The organization must provide extension developers with procedures and tools for extension testing. Also, the organization must provide typical textual test scenarios and may ask developers to submit their test cases for future extension certification. The test procedures are deeply rooted in the quality management process for the platform and provide developers with insight into the qualities that the keystone values.		
When implemented: • The test manager approves of test procedures for extension developers. • Tool support is provided to enable extension developers to test their extensions (semi-)automatically.		
Role responsible: Quality manager		***Literature***: [29, 30, 9]

based on APIs and SDKs, and the constant listening to the requirements and feedback from extenders in the field.

- **Software Quality Manager -** Software quality managers must realize that with the advent of software ecosystems, they become responsible for the quality and security of the product beyond the scope of the company. Extenders, often less equipped to provide the same levels of quality and security, must be guided to make sure they do not accidentally introduce vulnerabilities into the ecosystem [8].

- **Software Release Manager -** Software release managers are responsible for the correct and complete delivery of new software versions to extenders and customers. However, to avoid lag, quality problems, and extension incompatibility, software release managers must coordinate early releases to extenders, to provide them the opportunity to test and develop their extensions against the newer versions of the platform [33].

- **Community Manager -** The community manager becomes responsible for managing the different communities, whether it is the comprehensive partner community (f.i., all businesses developing apps for Android), or separate developer communities (f.i., the Chrome extension developer community). The community manager needs to constantly be aware of the developments in the community, needs to listen to developers, and make sure that software developers in the ecosystem are happy and productive. The community manager does so by maintaining community portals, organizing developer meetings, and directing developer feedback to the product manager and architects [34].

- **Partner Manager -** The Partner Manager is responsible for all commercial interests of partners. The Partner Manager is creating business opportunities for partners, to enable them to create value for themselves and the ecosystem. As such, the Partner Manager is concerned with creating partner models [18], enabling different business models, and connecting potential customers with partners.

- **Support Manager -** The responsibilities of a support manager within a software producing organization are suddenly expanded when an ecosystem with developers starts gathering around an organization. The support manager becomes responsible for providing answers, training, and documentation for extension developers, who's questions are of a different nature than those of customers.

Hess et al. [35], provide a list of 'new' roles in an organization who enable ecosystems, such as "global ecosystem evangelist". The role above has been kept deliberately traditional, as to find the best fit with the current organization of a software producing organization.

4. The Case Studies

In this Section experiences with the SEG-M^2 are shared. We describe the empirical case studies that followed the case protocol, alphabetically.

4.1. ARCompP1

Description: ARcompP1 is a platform for Augmented Reality that provides Application Programming Interfaces (API) in C++, Java, Objective-C, and the .Net languages through an extension of the Unity game engine. With the use of 2D and 3D targets, augmented reality provides a new way to perceive the environment around combining virtual to real. ArCompP1 was introduced five years ago and has become an industry-leading platform on which all kinds of international companies build extensions. In 2015 it supported a global ecosystem of 175,000+ registered developers and has powered 20,000+ apps with more than 200 million app installs worldwide.
Orchestrator Motivation: ArCompP1 has been developed for developers wanting to create augmented reality solutions. The revenue model depends on the number of developers that use the platform in their applications, providing abundant reason to create a healthy ecosystem. The ecosystem has been built up in a similar manner as the Unity platform ecosystem, on which ArCompP1 depends.
Maturity Levels: Surprisingly, especially when looking at the high number of developers, ARcomp has rather immature governance of its ecosystem. First, it does not

Table 3: The SEG-M^2. Dev stands for developer, Devm for development. Part 1 of 2.

1	*Associate Models*	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1.1	**Partner promotion and grooming**		Scout strategic partners	Partner relationship model	Partner training and showcasing	Certification	Partner health analysis	Involve start-ups	Partner exclusion
1.2	**Partnership management**		Informal agreements	Partner contracts	Associate model			Advanced associate model	
1.3	**Consulting partner support**			Informal consultancy partner support	Formal trainings		Consultant certification	Organizing consultant events	
1.4	**Connect customers and partners**		Direct customers to partners		Create partner index	Provide ticketing system	Provide customer contact data to partners		Share customer configurations
1.5	**Marketing and sales**		Partner and customer focus			Co-acquisition	Revenue sharing	Partner focus	
1.6	**Training**			Simple getting started guides	Prof. training organization	Certification based on training		Partner employee management	
1.7	**Sales partner support**		Informal sales partner support		Certify sales partners	Market-specific sales groups	Organizing local sales events		Partner awards
2	*Sw. Devm. Governance*	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
2.1	**App testing**		Informal tests	Create app test procedure	Share app test procedure	Binary ext. test proc	Allow partners to self-test		Partners submit tests with App
2.2	**Application quality**		Support partners		Platform sandbox	Detect quality issues	Share issues with partners		Create SOK portals
2.3	**Dev relationships**		Informal contacts	Dev. meetups are organized	Coordinated feedback channels	Dev. interaction is supported		Partners help partners	Devs contribute to other devs
2.4	**Devm process automation**			Quick install for SDK or streamlined API adoption		Automated testing	IDE extensions	Automated releasing	
2.5	**Devm support**		Informal dev partner support	Dedicated engineers	Knowledge infrastructure	Ticketing systems	Collaborative road mapping	Collaborative dev.	Facilitate ecosystem of ecosystems
2.6	**Requirements sharing**		Informal transparency	Formal communication policy	Requirements portal	Devs role in requirements portal	Partner supports prioritization		Partners pick up requirements as co-devs
2.7	**Roadmapping**			Open roadmap			Partner extensions taken into account	Partner extensions part of road map	
2.8	**Dev monitoring**		Informal monitoring	Monitor feedback channels for dev motives		Document dev wants and needs	Adjust docs according to demands	Study dev behavior through SOK	Use automatic data collection from IDE
3	*Open Markets*	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
3.1	**app stores, component markets**		Internal extensions list		List of extensions		app store	Microservice architecture	Dynamic app composition
3.2	**Application format and delivery**			Integrateable components, manual installs		One-click install of integrations	On-demand applications	Extendable applications	
3.3	**App approval process**				Informal	Establish app approval team	Process support and automation	Self-moderation by end-users	External partners approve apps
3.4	**App curation**				Opportunistic	Formal ruleset		Appeals policy	Community curation support
3.5	**App marketing**						Marketing of extensions in app store	Marketing of extensions outside of app store	
3.6	**Community Engagement**				Create dev forum		Organize dev-cons and hackathons	Showcase devs and solutions	Showcase tools by devs
3.7	**Business model innovation**			Reseller model		app store model	In-app purchases		Subscription

Table 4: The SEG–M^3. Dev stands for developer, Devm for development. Part 2 of 2.

4	*Intellectual Property*	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
4.1	**Licensing**		Local products licensed					Sharing licenses with partners	Automated checking of license violations
4.2	**Digital asset management**		Reuse policy for internal products	Reuse policy for external products	Reuse policy for internal products with partners	Reuse policy for external products with partners			Contributions to other ecosystems coordinated
4.3	**Patent management**					Third party patents licensed	Patents created for the platform	IP sharing with partners	Patent violations identified

5	*Open Platforms*	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
5.1	**Platform hardening**			First weaknesses identified	Guards built in	Structural hardening process	Architecture becomes first class citizen		
5.2	**Platform extensibility**		SDK or API	Multi-layered ext. framework		IDE Support			Fourth party extensions
5.3	**Software operation knowledge**			Platform SOK gathered		App SOK gathered	Sharing bugs and crashes	Sharing usage	Sharing customer configurations
5.4	**Platform doc.**		Doc. with getting started	Doc. with examples	Doc. generated from code	Interactive documentation	Feedback gathered	Prioritization based on knowledge needs	
5.5	**Security**			Security scans	Security policies		Security policies shared with partners	Security certification of partner components	Security alerts shared in ecosystem
5.6	**Platform evolution**					Evolution policy established			Directed feedback to partners about platform use

6	*Ecosystem Health*	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
6.1	**Competing ecosystem analysis**		Informal competition analysis		Reference competitors developed		Policy for contributing to other ecosystems	Domain engineering and niche discovery	Partners guided in contributions to other ecosystems
6.2	**Market and customer analysis**		Market analysis for platform		Market data shared		Customer surveys	Automated data collection	Customer data shared
6.3	**Partner health assessment**			Ask partners for performance data	Strategic partner analysis			Partner surveys	

7	*Open Innovation*	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
7.1	**Standards participation**			Standard adoption		Participation in standard bodies			Creation of new standards
7.2	**Partnering with academia**				Academic contacts		Collaboration in research projects		Shared R&D center
7.3	**Dev inspiration**			Stimulate in-company innovation	Promote partner solutions	Show partner innovations to partners		Reward new innovations	
7.4	**Open technology road maps**			Informal sharing		Formal presentation			Collaborative road maps

provide many facilities for its developers, such as ticketing, testing tools, or a platform sandbox. Primarily, developers use the forum for reporting problems and platform wishes.
Evaluation Findings: One particular reason why ARcomp perhaps does not invest in the ecosystem extensively, is that ArCompP1 largely depends on Unity, a platform that is highly mature. We find that Unity provides many of the services of which ArCompP1 profits. In a sense, ArCompP1 parasitically lives on the Unity platform, while simultaneously supplying new customers for Unity. Also, as ArCompP1 is a development tool, ARcomp does relatively little to curate its ecosystem and manage it. After all, when developers pay to use ArCompP1, they self-select on being invested in the ecosystem and creating valuable extensions. Furthermore, these apps are released in dedicated app stores, which by themselves provide curation.
Reflection on the Model: The main reflection from the ARCompP1 case study is that it is hard to deal with a platform that is in itself part of another platform. Effectively, some of the advice that would follow from the model for ARCompP1 is easily countered with "that has already been implemented in Unity." Such practices were not checked off, however, as being part of another platform relinquishes control to another party.

4.2. ERPCompP1

Description: ERPCompP1 is a Dutch company that currently has around 200,000 enterprise resource planning customers worldwide with approximately 2,000 employees. They have an on-line product that is popular, in large part due to the availability of an API and an app store with hundreds of connecting apps. These apps are mostly built by small independent software vendors that offer services that can benefit from connecting to an enterprise resource planning package.

Orchestrator Motivation: ERPCompP1 was always relatively strong in managing its network of resellers, but, as they converted their customers from stand-alone to Software as a Service (SaaS) subscriptions, the role for resellers was diminished. ERPCompP1 provided the resellers a chance to become partners by developing extensions to ERPCompP1's products. Besides, ERPCompP1's management realized that their product is central to any company, and in its role as an information hub could become much more relevant by connecting to third party services. ERPCompP1 has collected data on its customers that show that customers that use more than one third party extension are 20% less likely to transfer to a competitor.
Maturity Levels: The organization is currently at low levels of maturity, even though it scores considerably higher on several of the seven domains. ERPCompP1 falls short on some of the practices. First and foremost, customers are still seen as the main focus for the company. Apps are a relevant source of revenue, but customers still contribute significantly more to the bottom line.

Currently, ERPCompP1 is governing API use for its partners, but without too strict limitations. There are no test procedures, no release schedules, there is no IDE support, and little operation data is gathered. Also, the ticketing system (bugs and feature requests) has not been opened up to partners. ERPCompP1 is not secretive, but the system they are using simply does not allow for opening up the bug tracker to the outside world. Most of these have not been implemented due to the age of the API (2 years old) and the fact that ERPCompP1 is still figuring out the most profitable way to deal with its partners.
Evaluation Findings: The ERPCompP1 ecosystem is a success story. Many of the partners are enthusiastic about the platform, mostly because the platform enabled them to grow rapidly. Some partners even managed to piggyback internationally, to some of ERPCompP1's international customers. ERPCompP1 is constantly growing its ecosystem and trying new strategies. Presently it is experimenting, for instance, with a percentage fee of revenues made through the app store, although it is significantly lower than Apple's 30%, which is seen as a benchmark in the industry. The revenues made with apps in ERPCompP1's app store are made in many varying ways: some companies offer a pay-per-use model, whereas others offer a pay-per-month model. ERPCompP1 has made separate agreements with each of its customers, which, although not scalable, has been a worthwhile experience in discovering the opportunities in the ecosystem with partners.
Reflection on the Model: The managers at ERPCompP1 evaluated the maturity model for software ecosystem management positively. It was considered a welcome contribution and highly informative. They considered the collection of practices highly useful. The maturity ranking was seen as an interesting guideline, but they considered that they could cherry pick the most valuable practices. When benchmarked with other platforms (Microsoft CRM, SalesForce), managers at ERPCompP1 considered it obvious that other organizations were more mature; mostly because these organizations had more strategically invested in the ecosystem. Two new research challenges were introduced by ERPCompP1: (1) providing customer data to third parties (data governance) was considered to be a research challenge on its own and (2) finding the optimum business model for what is essentially a data platform is challenging too.

4.3. CreditCompP1

Description: CreditCompP1 is currently the leader in its market of credit management. They realized that their traditional reseller partnerships are no longer sufficient for continued growth and increased revenues. A new strategic directive has been set to increase the level and maturity of the strategic alliances with external entities, both existing as well as yet unexplored opportunities. Illustrated by Director Channel Development: *"our number one priority is how to systematically attract new innovative partners to our ecosystem"*.

Orchestrator Motivation: CreditCompP1 realizes that their only way to grow larger, is to start forging relationships with extenders and identifies this as a main priority for the next five years. Their platform is immature and they are currently redesigning their architecture to enable more third party extensions. There is also a strong market drive, as some of the competitors are performing better in terms of financially successful partnerships. The company has implemented many practices and is relatively mature in terms of software ecosystem governance. Their software ecosystem governance processes are well developed, indicated for instance by a well implemented set of practices for ecosystem health management, but worryingly, they perform these practices on a low number of partners.

Maturity Levels: CreditCompP1 is struggling in two areas within their propriety ecosystem, being collaborative software development, and software ecosystem growth. In terms of collaborative development, besides outsourcing one third of their in-house software development to two partner companies, there is little third-party development for their lead product. The first problem is directly related to their multi-channel software platform, on which the the opportunities for extensions and independent development are built. In the words of its CEO: *"the architecture of our product is like spaghetti,[...], making the transition to an extendable multi-tenancy cloud-based platform extremely difficult due to many dependencies built over the last 23 years"*.

Evaluation Findings: The second problem is found in the financial performance of the partner portfolio. The sales director illustrates the financial implications of an unbalanced and poorly designed partner portfolio: *"over the last 3 years, only 2 out of the 12 business partnerships yielded positive financial returns"*. This is supported in the words of CreditCompP1's CEO: *"to become best-of-breed credit management software provider, our partner portfolio must grow exponentially. Not only grow, we need to rethink our partnership approach"*. Although the recent incentive to implement an extended associate model to support four clusters (referral, reseller, integrator, white-label) should increase the portfolio diversity, the partner alliance manager reveals that this might not be enough: *"our partner selection process is mostly based on trust. There is no step-by-step approach that we follow for every single prospect"*. Analysis of the four associate models further revealed the lack of balance in entry criteria and missing partnership value analysis.

Reflection on the Model: One of the main findings from the evaluation with CreditCompP1 was that they were enthusiastic about the model, but realized that the size of their ecosystem, currently at around 30 partners, does not yet warrant such a heavy investment in ecosystem governance. They discovered throughout the evaluation that the vision in the company has been too much technological and too little on the actual attraction of new partners.

4.4. NetCompP1

Description: NetComp is a large Asian network equipment manufacturer with a large product portfolio. The company has been active for decades as a prize fighter in the market, but has reached the status of being a household brand. As their equipment is pervasive in the market, there are many ecosystem opportunities that are presently being pursued by NetComp. We have evaluated three different platforms within the NetComp company, NetCompP1, NetCompP2, and NetCompP3.

Orchestrator Motivation (NetCompP1): NetComp is active as a mobile manufacturer and mimics the competition in terms of the surrounding supporting ecosystem functions. As such, they provide a developer SDK for mobile apps, error reporting services, a dedicated app store, business models for app developers, etc. The software ecosystem governance practices are relatively mature and well developed, in large part because the organization has set itself a goal to compete with some of the largest mobile manufacturers internationally. The main motivation for evaluating the software ecosystem governance for NetCompP1 is to ensure that no practices are being ignored and that they do everything in their power to support app developers.

Maturity Levels: One interesting observation within NetCompP1 was that a dedicated Integrated Development Environment (IDE) was created for app developers. The main reason for doing so was to ensure that app developers would be able to create apps easily. A secondary reason was to ensure that app developers use secondary features from other products and platforms at NetCompP1. An app developer can, for instance, by default use NetComp's extensive cloud offerings and billing solutions through libraries that are readily available in the SDK. Implementing cloud offerings from another company would be possible, but as the IDE comes with such features integrated, it is tempting for developers to simply use what is available from NetComp.

Evaluation Findings: As the large mobile manufacturers were mimicked, the team behind P1 was not that impressed with the model: they had themselves already implemented most of the practices and found it a useful overview.

Reflection on the Model: In the discussions with the team, it was found that while the model was considered useful, they were far more interested in in-depth tool evaluations of tools used by large mobile manufacturers. Tools for billing, user tracking, developer tracking, and for example testing, were all considered relevant. Recently, we published the findings from a study that was the result of our work with NetCompP1 [36].

4.5. NetCompP2

Description: NetCompP2 is a platform that was specifically designed to facilitate the ecosystem of NetComp, as discussed in earlier work [31]. Several hundred products

were included in the initiative: NetComp decided that all products must unify their partner acquisition and external developer relations in one platform. The challenge of introducing NetCompP2, by now relatively successful, has been both a managerial one and a technical one. Many of the departments managing the products affected this way had already been building up ecosystems and platforms of their own, so a cultural change was needed to accomplish that all these products would start growing their ecosystems in a similar way. An advantage has been that in this way the knowledge of platform and partner management was unified in one team, which is relatively unique for such a large firm. We highlight several discussion points experienced by NetCompP2 in their ecosystems initiative.

Orchestrator Motivation (NetCompP2): NetComp realizes that it has been successful at growing their company autonomously, but also realizes that it can grow revenues without growing its employee base, by mobilizing its partners more. This incentive has led NetComp to undertake a company wide ecosystem improvement program. It was also in part to eliminate the redundancy of each product unit developing their own partner and third–party developer management systems.

Maturity Levels: In the communications industry security is a major concern. NetCompP2 architects are responsible for executing and checking security guidelines. These guidelines are well documented and well managed in NetComp. The architects have three levels of security check in place, which we cannot share for reasons of confidentiality. However, we are allowed to illustrate some of the guidelines that are used by the architects. At the first level, the architects look at data leaks, unlawful interception, and privacy protection. At the second level, the architects have more advanced steps, such as data encryption, attack and integrity protection, and log auditing. At the third level, the architects apply tools such as virus protection, security hardening, protected installations, database hardening, and some guidelines for partners on security. An interesting observation is that NetComp presently shares little of this knowledge with partners, whereas partners can greatly benefit from security audits. There are many ecosystem opportunities here: partners can be audited, certified, and trained in the domain of security. NetComp is evaluating these different options presently.

Evaluation Findings: As the hardware running for customers is generally deployed and then left alone, so are the NetComp products. This results in situations where the NetComp products running on extensible hardware is running far behind the most recent version, making it harder to develop against. It is, however, a challenge to convince partners to update the software running on the hardware and its accompanying Netcomp servers without any business incentive. Simultaneously, however, when a customer wishes to acquire extended features through a NetComp partner, all hardware drivers must first be brought up to date. NetCompP2 is working on a policy to incentivize partners to upgrade software, even when there is no direct need for the partner to do so.

Reflection on the Model: NetCompP2 reflected on the model highly positively and the team behind it still uses it to set goals and improvement projects for the future. NetCompP2 was evaluated twice, where in the second phase many of the prescribed practices were implemented by NetComp. NetComp had made significant progress in implementing the practices. The managers at NetComp indicated that some practices proved to be much harder to implement than initially expected. Implementing a requirements management system that is also open to third parties, for instance, would face both technical and cultural problems within the organization. In NetComp, opening up the requirements management system to partners has proven so challenging, that the department managers have adopted the system of a third party[1] that interfaces with the internally used system through an API and through manual data copying.

The main practices that were implemented between the two evaluations were related to partner empowerment. Partners are now trained, certified, and their solutions can be showcased. Furthermore, partners can now test their extensions in sandboxes, get detailed testing procedures for their apps, and can prioritize bugs and features in bug tracking systems. Thirdly, app stores and extension lists are being created for the different platforms, to enable partners to sell solutions directly to NetComp customers. Finally, developers are now informed of a widely known release schedule, as to preserve compatibility over different versions. A broader discussion on the use of the model at NetCompP2 is given in Section 6.2.

4.6. NetCompP3

Description: NetComp is a supplier to many wireless telecommunication providers (TelCos). NetComp provides hardware to create the infrastructure that is needed to enable wireless networking. The TelCos are seeing their roles diminish, as increasing numbers of customers are only using their services to connect to the internet. Whereas in the past TelCos could charge for services related to text messaging, phone minutes, international calls, content services, they are now being pushed down the stack, meaning that they are forced to take on a role as an infrastructure provider: an activity that is perceived as less profitable.

Many of the telecommunication providers are trying to offer extra services by diversifying into different domains. NetComp is supporting these organizations by providing them with a business to business platform (NetCompP3) that enables telecommunication providers to build their own ecosystems.

Orchestrator Motivation: NetComp was challenged by its TelCo customers to support them in developing new business models and new ways of engaging the TelCo clients. NetComp has been somewhat reluctant to develop

[1] Uservoice.com

its own ecosystem around NetCompP3, because NetComp feels it is competing with the TelCos if it enters that business too deeply. On the other hand, they can offer their TelCo customers with a network of relevant add-on service providers that may be relevant to their region. NetComp has been experiencing this as a tightrope, but its TelCo customers are excited about the new opportunities the TelCos can offer their clients.

Maturity Levels: NetCompP3 scored the lowest of all cases. The main reason for this, however, was simply that this type of business is new to NetComp and many of the features that were not directly implemented in the platform itself, were later dedicatedly implemented and customized for the TelCo customers. NetCompP3 was only just discovering that many of the features that were built customly for customers may also be relevant for other customers and is currently going through a transition from product to product line [37].

Evaluation Findings: NetComp plays an interesting role here. In part it provides the technology for the TelCos to build their own ecosystems, but simultaneously NetComp is itself finding partners for the TelCos and sharing them amongst the TelCos. NetCompP3 is effectively designing an ecosystem of ecosystems.

Reflection on the Model: During the case study, we identified that NetCompP3 has two ecosystem perspectives: the TelCo's ecosystem and NetCompP3's ecosystem. For this study we focused on NetCompP3's ecosystem. At NetComp, however, a proposal was made to support the TelCos with a maturity model of their own, that mainly focused on the lower levels of the SEG-M^2.

5. Benchmarking the Software Ecosystem Orchestrators

Table 6 provides a benchmark for the different products that were evaluated in the case studies. The table shows which share of the practices has been implemented at a case company, out of all available practices. These percentages are misleading, as some practices may encompass a much larger amount of work than others. The percentages do show that there are relationships between the different focus areas, i.e., if an organization scores well in one aspect, they will probably also score well on another. We also find that two platforms and their organizations score high: Android and iOS. These referential ecosystems have grown by focusing on providing a high quality platform, a high quality product, and an active healthy ecosystem.

Several observations can be made when studying average and standard deviations per process area in Table 6. First, we notice that roadmapping, licensing, and app marketing have the lowest standard deviations, i.e., most organizations score around the same in these areas. For each of these process areas it is relatively easy to reach the first practice but relatively hard to implement higher level practices. For licensing, for instance, the lowest level practice immediately brings an organization to level 5, simply by licensing its own products, something that practically all software producing organizations do.

When looking at standard deviations, it can be observed that some practices show far less consistency. For example, "Consulting partner support" is a practice that some companies have perfected, such as Microsoft and CISCO, while others, such as Eclipse and NetComp have not focused on at all and thereby score significantly worse. The same holds for "software operation knowledge", as some companies collect large amounts of information about their partners and extenders, whereas others do not.

We also find some extremes, where where the average scores are relatively low or high. For instance, very few of the platform orchestrators enable extenders frameworks for self-testing of extensions or provide them with instructions on how to make sure the extension works well on the platform. Another example is platform documentation, where typically, organizations keep a relatively simple set of documents to describe the extension procedure, but have relatively little interactivity or optimization in this documentation. On the high side we find "marketing and sales" activities and "community engagement". It becomes obvious that all platform orchestrators in this study find collaborative marketing and sales with partners a priority.

These generic observations are indicative that the model differentiates between different platform governance strategies and that there are less and more developed areas in the model. The less developed areas present opportunities for future research. Furthermore, the averages for each focus areas all lie around 60% which, even though not statistically proven, indicate internal consistency of the model.

5.1. Experiences with the Benchmark

During the case studies the organizations were highly interested in the benchmark table, especially because it gave them an indication of what others achieved and why. One of the most often heard comments was, when seeing this table, that the organization still had a long way to go, and wanted to be equal level as another ecosystem in the top 4.

The fact that the higher scoring platforms are nearing 100% indicates that it is unclear from the literature what the next steps for these ecosystems are. We dare speculate that once an ecosystem is optimally working, i.e., has achieved the highest level of management maturity and also penetration of potential partners, it needs to diversify. We observe that iOS, for example, is venturing and specializing in more domains, such as health. Furthermore, these platforms become fundamental pillars in our technology stack, and strong as they are, will soon be overgrown by other platforms, such as ambient ubiquitous voice interfaces, augmented reality platforms, etc.

5.2. Motivations for Software Ecosystem Orchestrators

We have also asked the ecosystem coordinators what their motivation is for growing an ecosystem and what their motivation is to improve the ecosystem. The results of these questions are found in Table 5. The reasons for growing an ecosystem are varied, but some patterns can be recognized. First, organizations that started as product vendors indicate that platformization is imminent: their product is becoming increasingly vital to the business and increasingly is built upon by partners and customers themselves. A second recurring motivation is the extraction of value from the ecosystem, for instance by using app stores or by using a pay-per-use fee for platform extenders. These findings resonate with the motivators identified in the introduction of this work in Section 1.1.

The motivations for improving the management of the ecosystem are myriad as well, although trends can be discovered. One of the main indications from platform owners is that they see that the competition is doing similar things and is doing better or improving rapidly. The second motivation for management improvement is that customers and partners complain of insufficient extensibility, insufficient transparency about development, and a business model that is not profitable for them. Partner growth is also a main driver for improvement of the ecosystem's management.

6. Discussion

6.1. Model Validity

The evaluations led to several interesting findings about the model validity. Unintentionally, the average of all practices is around 65%. While the actual value is not relevant, what is relevant is that there appears to be internal consistency in the model, i.e., none of the focus areas have outlying scores. Secondly, during the interviews few discussions arose about whether the practices were inappropriately placed; this discussion was typically avoided by marking a practice as irrelevant for the particular case. There were no patterns in which practices were marked as irrelevant and typically 3-5% of practices were marked so.

In the process of applying the SEG-M^2 it was found that there are varieties in ecosystem and platform type. More specifically we found that a platform that is extended through SDKs can vary greatly from, for instance, an API platform or a mobile (app) platform. There are several solutions to this problem, but they have been cast aside. First, different models could have been developed, such as the SEG-M^2-SDKs versus SEG-M^2-APIs. This is not elegant, however, and makes the SEG-M^2 less maintainable. A second solution, which would have been favorable from an academic standpoint, is to abstract away such practices to higher level practices. Initially, for instance, the SEG-M^2 did not include the app store practice (*practice 3.7.4*), but instead included a more abstract "open extension market" practice. Practitioners, however, objected to this term during the evaluations, so pragmatism was chosen over elegance. The mechanism that can be used for solving this problem, is by simply marking a practice as not-relevant for the organization. Interestingly, however, is the fact that even though one could classify the Salesforce.com platform as an API platform, it has an app store as well, and the same holds for the case of *DutchSaaS*.

The marking of a practice as "irrelevant" is a direct threat to the quality of the SEG-M^2. In the evaluation of *NetCompP3*, the organization initially asked for all non-technical practices to be marked as unnecessary. This actually revealed an interesting split in the SEG-M^2, and in the organization as well. Within the SEG-M^2 it shows that the SEG-M^2 is split over both 'commercial' and technical concerns, as is shown in Figure 3. Secondly, it shows an organizational split in NetComp that may even be harmful to the software ecosystem initiative.

The first version of the SEG-M^2 was created in 2014. The SEG-M^2 has continued to evolve quite significantly since that first moment and determining the right time for publication has been challenging: should we wait longer until the SEG-M^2 stabilizes, or its evolution be accepted and dub the version published here as version 1? We have chosen the latter but expect the SEG-M^2 to continue to evolve. The main reasons for this are technological evolution (is the era of the app store almost over [39]?) and the introduction of new practices (will grey-out tests for APIs become standard in the industry? [40]).

Another relevant question posed by the case participants is the role of metrics for the SEG-M^2. After all, using key performance indicators about software ecosystem health [41], could indicate how organizations can practically improve their ecosystem health over time. At the time of writing, however, no clear relationship has been established between the practices and their potential effects on these ecosystem health indicators, which is taken as a direction for future work.

6.2. Third Party Evaluations

We provide third party evaluators with the following tools. First, a full description is provided [13] for each of the practices, including a number of conditions that need to be true before one can evaluate the practice to be fully implemented. Second, we recommend that the organization under study is first evaluated, before a maturity ambition level is established, to make sure the evaluators are not biased when entering into the process. Finally, the recommendation is that the evaluator backs up every practice with proof, whether it is a web site, a document, or a code fragment. A question of validity is whether the assessment can only be conducted by the researchers who developed the SEG-M^2, or whether it can be adopted by third parties. We are happy to report that an independent use of the model at a company led to "relevant improvement points" and "enables us to follow a structured improvement approach". One of the main challenges for the independent company was to judge whether a practice

Table 5: Motivations for Ecosystem and Management Improvement. The platform owners are mostly driven by opportunities (value extraction) and fear (competition is doing better). Organizations that provide core features (e.g., ERP, cash flow management, telephone services) identify that there business is being pushed lower down the stack, so platformization is inevitable. "# Interv." indicates the number of formal interviews conducted at the case company. The "# Days" indicates the number of days spent in the case company setting.

Organization	**Motivations for ecosystem**	**Motivation for ecosystem management improvement**	**Platform technology offered as**	**# Interv.**	**# Days**
ARCompP1	Platform, started as development library Requires co-innovation Extract value from the ecosystem Strategic position within Unity ecosystem	Recent change of ownership Competition is catching up	Development library	6	5
ERPCompP1	Primarily aimed at SMEs with varying needs Extract value from ecosystem Platformization imminent	Competition is doing better Partners complain about business model Ecosystem is used to convince customers	SaaS	5	5
CreditCompP1	Partners want to extend the platform Customers say value of platform improves Platformization imminent	Attract high quality, high value partners Value extraction from ecosystem	SaaS	8	7
NetCompP1	Combination of different platforms One partner management system One ecosystem	Partners complain company is too closed Insufficient numbers of partners Competition is doing better	Development library and Set of APIs	17	10
NetCompP2	Extract value from the ecosystem Connection to other platforms Network effects	Attract more developers and ISVs Competition is doing better	Operating System	10	6
NetCompP3	Platformization imminent Customers need to innovate API library is extra source of income	APIs are poorly documented Creation of customer ecosystem Extract more value from ecosystem Innovation too slow acc. to partners/customers	Product with APIs	8	5

	Focus Area	Android	iOS	SalesForce	Microsoft CRM	Cisco	NetCompP1	Eclipse	NetCompP2	CreditCompP1	ARCompP1	ERPCompP1	NetCompP3	Avg	Stdev
1	Associate Models	92%	96%	86%	100%	96%	55%	47%	49%	86%	71%	47%	29%	67%	
1.1	Partner grooming	6	7	6	7	7	2	3	3	6	3	2	2	4.1	2.13
1.2	Partnerships	7	7	7	7	7	5	5	2	5	5	5	2	5.0	1.83
1.3	Consulting partner support	7	7	7	7	7	1	4	5	7	7	7	1	5.3	2.50
1.4	Connect customers and partners	6	7	3	7	7	7	3	3	6	3	3	2	4.4	2.07
1.5	Marketing and sales	7	7	7	7	7	7	3	3	7	7	1	5	5.4	2.27
1.6	Training	5	5	5	7	5	3	3	5	5	3	3	2	4.1	1.52
1.7	Sales partner support	7	7	7	7	7	2	2	3	6	7	2	0	4.3	2.75
2	Soft. Dev. Governance	88%	86%	82%	63%	73%	66%	64%	64%	52%	41%	38%	43%	59%	
2.1	App testing	6	4	6	4	6	6	1	4	4	1	1	1	3.4	2.22
2.2	Application quality	7	7	7	6	7	7	3	3	3	2	3	6	4.7	2.06
2.3	Developer relationships	7	7	7	5	3	3	7	6	2	6	3	6	4.8	1.87
2.4	Process automation	5	7	5	5	4	5	5	5	4	3	3	1	4.0	1.33
2.5	Development partner support	7	5	7	6	7	4	6	4	5	4	4	4	5.1	1.29
2.6	Requirements sharing	6	6	3	4	6	3	7	6	4	2	1	1	3.7	2.11
2.7	Roadmapping	5	5	4	5	4	4	4	4	4	1	4	4	3.8	1.03
2.8	Developer monitoring	6	7	7	0	4	5	3	4	3	4	2	1	3.3	2.00
3	Open Markets	88%	92%	92%	61%	59%	78%	57%	65%	49%	49%	63%	41%	61%	
3.1	App market	6	5	6	4	5	5	4	5	2	4	6	2	4.3	1.42
3.2	Application format and delivery	5	5	7	4	3	4	3	4	3	3	4	3	3.8	1.23
3.3	App approval process	6	7	5	3	3	6	3	5	2	2	5	3	3.7	1.42
3.4	App curation	5	7	6	5	5	5	3	5	5	2	5	3	4.4	1.26
3.5	App marketing	7	7	7	5	4	5	5	4	4	4	5	3	4.6	1.07
3.6	Community Engagement	7	7	7	5	6	6	7	5	4	6	2	5	5.3	1.49
3.7	Business models	7	7	7	4	3	7	3	4	4	3	4	1	4.0	1.83
4	Intellectual Property	90%	90%	71%	67%	67%	52%	90%	57%	57%	52%	48%	62%	62%	
4.1	Licensing	6	5	5	5	5	5	6	5	6	5	5	5	5.2	0.42
4.2	Digital asset management	6	7	3	3	3	3	7	2	2	2	2	3	3.0	1.49
4.3	Patent management	7	7	7	6	6	3	6	5	4	4	3	5	4.9	1.37
5	Open Platforms	95%	86%	95%	83%	71%	79%	43%	71%	48%	48%	45%	38%	62%	
5.1	Platform hardening	7	7	7	4	7	7	3	7	3	7	3	3	5.1	2.02
5.2	Platform extensibility	7	6	6	6	6	6	3	6	3	3	3	3	4.5	1.58
5.3	Software operation knowledge	6	6	7	7	6	7	1	3	5	1	3	3	4.3	2.41
5.4	Platform documentation	7	5	6	4	4	4	3	3	2	2	3	3	3.4	1.17
5.5	Security	7	5	7	7	7	5	2	5	1	1	4	1	4.0	2.58
5.6	Platform evolution	6	7	7	7	0	4	6	6	6	6	3	3	4.8	2.25
6	Ecosystem Health	95%	95%	95%	86%	81%	86%	38%	38%	71%	43%	33%	29%	60%	
6.1	Competing ecosystem analysis	6	6	7	4	6	4	2	4	3	4	4	2	4.0	1.56
6.2	Market and customer analysis	7	7	6	7	4	7	5	2	5	4	2	2	4.4	1.96
6.3	Partner health assessment	7	7	7	7	7	7	1	2	7	1	1	2	4.2	2.97
7	Open Innovation	93%	96%	89%	82%	82%	64%	86%	54%	32%	39%	46%	39%	61%	
7.1	Standards participation	7	7	6	6	6	3	7	6	3	3	3	6	4.9	1.66
7.2	Partnering with academia	6	7	6	4	4	4	6	4	2	2	4	2	3.8	1.48
7.3	Inspiration for developers	7	7	7	7	7	5	5	2	1	3	3	0	4.0	2.58
7.4	Open technology road maps	6	6	6	6	6	6	6	3	3	3	3	3	4.5	1.58

Table 6: The results of the evaluations at the case studies. Please note that the percentages are calculated as "implemented practices" out of "total practices" in a particular scope. As the practices are not weighed, and some practices are more far-reaching than others, please use the percentages in this table only indicatively. ARCompP1 and NetCompP3 were discussed in full in previous work [38, 31].

can be ignored (is it situational? Is it truly irrelevant for an organization to move to the next level of maturity?), whether it has been implemented in full or only partially, or should one state that it has not been implemented at all?

In Figure 4 a screen shot is shown of an evaluation at NetCompP2 at two stages in its development. The boxes marked in blue were part of the improvement project of the developer program after the first evaluation. The organization decided to build its own app store, focus on training developers, improved the partner model, and started organizing more developer conferences. The organization has

Software Ecosystem Governance Maturity Model								
1 *Associate Models*	0	1	2	3	4	5	6	7
1.1 Partner grooming		Scout strategic partners	Partner relationship model	Partner training and showcasing	Certification	Partner health analysis	Involve start-ups	Partner exclusion
1.2 Partnerships		Informal agreements	Partner contracts	Associate model			Advanced associate model	
1.3 Consulting partner support			Informal consultancy partner support	Formal trainings		Consultant certification	Organizing consultant events	
1.4 Connect customers and partners		Direct customers to partners		Create partner index	Provide ticketing system	Provide customer contact data to partners		Share customer configurations
1.5 Marketing and sales		Partner and customer focus			Co-acquisition	Revenue sharing	Partner focus	
1.6 Training			Simple getting started guides	Professional training organization	Certification based on training		Partner employee management	
1.7 Sales partner support		Informal sales partner support		Formalized sales partner support	Market-specific sales groups	Organizing local sales events		Partner awards
2 *Software Development Governance*	0	1	2	3	4	5	6	7
2.1 App testing		Informal tests		Create app test procedure	Share app test procedure	Binary application test procedure	Allow partners to self-test	Partners submit tests with App
2.2 Application quality		Support partners		Platform sandbox	Detect quality issues	Share issues with partners		Create operation knowledge portals
2.3 Developer relationships		Informal contacts	Developer meetups are organized	Coordinated feedback channels	Developer interaction is supported		Partners help partners	Developers can contribute to other developers
2.4 Process automation			Quick install for SDK or streamlined API	adoption	Automated testing	IDE extensions	Automated releasing	
2.5 Development partner support		Informal dev partner support	Dedicated engineers	Knowledge infrastructure	Ticketing systems	Collaborative roadmapping	Collaborative development	Facilitate ecosystem of ecosystems
2.6 Requirements sharing		Informal transparency	Formal communication policy	Requirements portal	Partner plays part in requirements portal	Partner supports prioritization		Partners pick up requirements as co-developers
2.7 Roadmapping			Open roadmap			Partner extensions taken into accoun	Partner extensions part of strategic road map	
2.8 Developer monitoring		Informal monitoring	Monitor feedback channels for developer	motives	Document developer wants and needs	Adjust documentation according to d	Study developer behavior through SOK	Use automatic data collection from IDE
3 *Open Markets*	0	1	2	3	4	5	6	7
3.1 App market		Internal extensions list		List of extensions		AppStore	Microservice architecture	Dynamic app composition
3.2 Application format and delivery			Integrateable components, manual installation		One-click install of integrations	On-demand applications	Extendable applications	
3.3 App approval process				Informal	Establish app approval team	Process support and automation	Self-regulation through app appraisal by e	App approval process with external partners
3.4 App curation				Opportunistic	Formal ruleset		Appeals policy	Community curation support
3.5 App marketing						Marketing of extensions in appstore	Marketing of extensions outside of appstore	
3.6 Community Engagement				Create developer forum		Organize dev. conferences and hacka	Showcase developers and solutions	Showcase libraries and SDKs from developers
3.7 Business models			Reseller model		AppStore model	In-app purchases		Subscription
4 *Intellectual Property*	0	1	2	3	4	5	6	7
4.1 Licensing		Local products licensed					Sharing licenses with partners	Automated checking of license violations
4.2 Digital asset management		Reuse policy for internal pro	Reuse policy for external products	Reuse policy for internal products with par	Reuse policy for external products with partners			Contributions to other ecosystems coordinated
4.3 Patent management					Third party patents licensed	Patents created for the platform	IP sharing with partners	Patent violations identified
5 *Open Platforms*	0	1	2	3	4	5	6	7
5.1 Platform hardening			First weaknesses identified	Guards built in	Structural hardening process	Architecture becomes first class citizen		
5.2 Platform extensibility		SDK or API	Multi-layered extension framework		IDE Support			Fourth party extensions
5.3 Software operation knowledge			SOK gathered about the platform		SOK gathered about App performance	Sharing bugs and crashes	Sharing usage	Sharing customer configurations
5.4 Platform documentation		Primitive documentation	Documentation with getting started	Documentation with examples	Prioritization based on knowledge needs	Feedback gathered	Documentation generated from code	Interactive documentation
5.5 Security			Security scans	Security policies		Security policies shared with partners	Security certification of partner components	Security alerts shared throughout ecosystem
5.6 Platform evolution					Evolution policy established			Directed feedback to partners about platform use
6 *Ecosystem Health*	0	1	2	3	4	5	6	7
6.1 Competing ecosystem analysis		Informal competition analysis		Reference competitors developed		Policy for contributing to other ecosy	Domain engineering and niche discovery	Partners guided in contributions to other ecosystems
6.2 Market and customer analysis		Market analysis for platform		Market data shared		Customer surveys	Automated data collection	Customer data shared
6.3 Partner health assessment			Ask partners for performance data	Strategic partner analysis			Partner surveys	
7 *Open Innovation*	0	1	2	3	4	5	6	7
7.1 Standards participation			Standard adoption		Participation in standard bodies			Creation of new standards
7.2 Partnering with academia				Academic contacts		Collaboration in research projects		Shared R&D center
7.3 Inspiration for developers				Stimulate in-company innovation	Promote partner solutions	Show partner innovations to partners	Reward new innovations	
7.4 Open technology road maps			Informal sharing		Formal presentation			Collaborative road maps

Figure 4: (not intended to be fully readable, the full model has already been given in Tables 3 and 4) Evaluation of NetCompP2, in two phases (2014, 2016). The practices marked with grey were implemented in the first evaluation in 2014. The practices marked in blue were observed to be implemented in 2016 after an improvement plan was followed that was created in 2014 after the first evaluation. Please note that the SEG-M^2 has evolved since 2014 as well, so some practices may be different from the final model presented in this article. The figure is not given to discuss the practices deeply, but to convey what kinds of results were gathered during the case studies; the full list of practices is found at the end of this article. Please also note, that an empty cell means that there is no practice in that cell and therefore organizations automatically fill those cells as though they had an accomplished practice in it.

made significant improvements, mostly because the managers of the organization wanted to satisfy partners in the ecosystem. The numerous improvements are appreciated by partners, who now feel that the organization is more committed to them than before.

6.3. The Role of Strategy

Wnuk et al. [42] show in the Axis case study that implemented governance mechanisms are subservient to organizations' strategy and business model. In this work, where the authors apply the governance model presented by Baars et al. [17] to a company that builds software embedded into IP cameras, they find that the organization at that point is no longer willing to further implement governance practices to improve the ecosystem. The company states that its ecosystem is currently large enough, represents too small a portion of the total revenue, and is currently investing more in innovating the products themselves. These decisions, although somehow conflicting with the ideas of a maturity model, are fully legitimate, as they simply represent a target maturity level that will not need to be improved on presently.

The Wnuk et al. case illustrates more than just a ambition maturity level, however. What it illustrates is that an organization will always need to make independent decisions when it comes to ecosystem growth. An organization can choose to employ the ecosystem as a way to grow larger, uncover new markets, and innovate faster. There are, however, other ways to do this, such as by focusing on other products in an organization or investing in particular in product innovation itself. As such, the SEG-M^2 should be seen as a way to move a company forward into the same direction, instead of a compass that changes an organization's direction altogether. In effect, that means that the SEG-M^2 still does not solve the big question of whether ecosystems are the best way forward for software producing organizations in this era. At best, the SEG-M^2 stimulates organizations to start thinking about monetizing the ecosystem, which is still a major strategic concern. We see the alignment of business models, technical architecture, and ecosystem design as future work, for ourselves and the software ecosystem community.

The operational practices given in the SEG-M^2 are relevant, but are overshadowed by two major concepts. First, an organization can be highly mature, while only having a handful of partners to collaborate with. Secondly, the SEG-M^2 is applicable on a low level mostly, while major decisions are being made on the platform portfolio level. It is highly relevant that Google enables its developers to find feedback mechanisms on Android, but it is perhaps even more relevant that Android integrates well with Search, Maps, and other highly successful platform products of Google. True platform success is defined by two other factors: (1) the total end-user and developer market share taken by the platform and (2) the complementary platforms that benefit the platform and create Staying Power [5]. As future work, we plan to continue down these roads in creating success indicators for ecosystems [41] and

further theory development of platform portfolio management [43].

6.4. The Role of Tooling

Throughout the case studies, we noticed there was significant interest into the tools that were used by different companies to support, enable, and manage their partners.

We divided these tools into four categories according to four phases of development. First, in the **initialization phase** we identify tools that enable developers to start development of a new extension to the platform, such as tools for platform documentation, platform sandboxes, and platform training. In the **development phase** we identify all tools that are designed to support developers in creating new extensions for a platform, such as IDEs, programming languages, collaboration tools, and testing tools. The third phase is the **deployment phase**, which consists of tools and services that enable developers to deploy and run their extensions, such as supporting cloud services (e.g., storage services), delivery infrastructures and app stores. Finally, in the fourth **'live extension' phase** tools enable developers to monitor, control, and monetize their extensions, using tools such as app stores, end-user analytics, and crash reporting tools. A full overview of such tools was reported in work with Baarsen [36].

6.5. Case Experience and Suitability of Maturity Models

The cases provided insight into how companies view the SEG-M^2 and use it as a guideline to further their maturity management. Typically two behaviors were observed: managers were tempted to just check the boxes and implement the practices as provided. Others, however, looked beyond the maturity levels and found practices at higher levels that matched their strategic goals and requirements. Overall, the interviewees stated that the model was presented in an understandable way and the practices were easy to interpret and build improvement projects around.

The cases also brought forward several reasons why a maturity model was an appropriate approach to present the large number of practices to organizations. First, organizations wanted to benchmark themselves against others in the particular domains. Even though that part has yet to be further developed, it proved highly effective to communicate that another organization had implemented a particular practice, and precisely how. Second, the maturity levels provided case participants with a feeling of achievement when moving up a level or gave them a call to action when they scored poorly. That said, there may be room for the interpretation of situational factors, i.e., organizational variety (e.g., open source versus closed source) that can determine the applicability of some of the practices. We consider the identification and interpretation of such situational factors [44] as future work.

One remark that must be made is that there exists insufficient tooling and theory around maturity models. We have chosen to create an article to disseminate the SEG-M^2, but we experience a serious lack of tooling in (for instance) presenting and detailing the practices. Also, the step from empirical evidence to checking off the practices in the maturity matrix, would have benefited from tool support to provide traceability between evidence and evaluated maturity level.

6.6. Theory Development and Hypotheses

Based on the work presented in this article, we develop the following hypotheses.

Focus Area Maturity Models are Useful for Dissemination of Complex Comprehensive Knowledge Frameworks. The evaluation from the case participants have illustrated that the focus area maturity model tool is useful for collecting knowledge about a focus area and the dissemination of it. The tool was chosen over a random list of practices, a decision that was supported by the case participants. In the future we would, if possible, again choose such models over flat lists of practices.

One of our largest challenges in this work was to develop a maturity model in a field that is rapidly developing, potentially introducing new domains, processes, and practices regularly. Interestingly enough, while there is a rapid increase of publications of new maturity models [45], there is little literature that particularly discusses the development of maturity models. The Maturity Model concept suffices, but there are definitely conceptual extensions possible that would make the creation of a maturity model easier. We propose two possibilities for future work. First, the maturity models could be extended with comprehensive version numbers. The changes over each version of the model should be elaborated to, for instance, calibrate any scores organizations have obtained in earlier models. Versioned models are a relatively common solution to this problem. A second possibility is the introduction of minor changes and major changes. A major version would be an official version that is approved by a governing body. Minor changes would be proposed as candidate changes and can be made continuously, such as the introduction of a new practice, or a change to an existing practice. These changes can then be taken into account with each assessment. The company would be scored as per the older certified version, while still seeing recent changes made to the model. We dub this method Extendable Focus Area Maturity Models.

Software Producing Organizations with higher levels of Ecosystem Governance Maturity have Well-Aligned Developer and Business Departments. Although this theory is highly anecdotal and should be classed as a hypothesis, it was obvious in the cases that having too much distance between the development and business departments led to slower uptake of the practices. Software ecosystem governance requires extensive collaboration between technical and business departments, whether it is the product management department, the sales department, the partner management department, or the marketing department.

Software Producing Organizations with Higher levels of Ecosystem Governance Maturity put Partners First. In some of the cases we noticed that a shift had taken place: customers were considered important, but partners were considered equally or even more important. As in these cases the partners generally represented a large contingent of the customer base, they were considered more important than some of the smaller customer groups. We hypothesize that when partners start representing large customer groups, their input will count equal to the input from customer groups. In some cases, where the software producing organization has a strong platform focus, the partners may become even more important than customers.

Open Source Platforms Manage their Ecosystems in the same way as Traditionally Closed Companies. The case of Eclipse introduces an interesting question: why have they chosen to develop a platform instead of just allowing new contributions to the platform from third parties? There are many reasons for this. The platform would be cluttered with third party contributions, the architecture allows for optionality and configurability, and partners can promote their own extensions. We hypothesize that such a platform architecture can be employed by any type of open source organization, whether it is governed by a community, a sponsor, a tolerant dictator, or a collective [46]. There are patterns between Eclipse's management versus the other platforms. They too experience the challenges of having to find new extenders and supporting them in their development endeavors. We therefore hypothesize that these large scale industry-friendly open source platforms are managed in the same way as traditionally closed companies manage their platforms. One could even argue that the most prevalent business models, such as SaaS, are no longer influenced by the openness of the code. Consequently, one can observe an increase in use of open source for strategic purposes, i.e., to attract more extenders and developers around a particular technology, rather than for strictly idealism and ethical coding. Furthermore, we conclude that as software producing organizations increasingly are growing towards each other, so will their software ecosystem governance practices. In earlier work we already concluded that software ecosystem development [47] is not significantly different for open and closed organizations.

7. Conclusion

This article presents the SEG-M^2, a maturity model for organizations that aim to improve the governance of their software ecosystems. The model is highly detailed and contains 168 practices that provide organizations with concrete, pragmatic, and implementable practices. The model has been created following maturity model creation steps of de Bruin [14]. The main sources for the practices are two structured literature reviews [2, 12] and six desk studies. The SEG-M^2 has been evaluated by assessing six SEG practices at four case companies. The evaluation shows that the SEG-M^2 provides an efficient way to improve an organization's capabilities in the domain of software ecosystem governance.

The article also shows insight into the application of the SEG-M^2 in practice, and provides a considerable set of empirical evidence from a set of international software companies, in the form of a benchmark. The overall findings from the benchmark is that large software organizations (Microsoft, Google, Apple, etc.) are extremely capable, whereas smaller companies with smaller budgets and markets can adopt practices from these large companies with ease, using the SEG-M^2. We also find confirmatory evidence that maturity models provide an appropriate mechanism for sharing complex sets of knowledge in novel fields.

During the research three new research challenges have been identified, in part with the participating organizations, which we are currently working on. First, one of the organizations established that they were managing a data platform as much as they were orchestrating a software ecosystem. We are now in the process of defining a focus area maturity model for data platform management. Furthermore, we are finalizing a study on partner management in the Dutch ERP industry, regarding the commercial processes surrounding partner management. Finally, we are developing a model for partner selection, i.e., the selection and attraction of the optimal partners with in a software ecosystem.

As part of our future work we hope to promote this model to consultancy firms, who can then apply the SEG-M^2 in practice and report on their experiences. Furthermore, we expect, as maturity models are always in development, to create a second version over the next years, based on feedback from more evaluations. The practices can be put into a wiki, to enable the community to add their experiences and comments to the wiki pages. We are in the process of developing a plugin for a content management system that is dedicated towards building focus area maturity models. One of the larger scientific challenges is to establish the effects of each of the practices on ecosystem health indicators [41]. One of the managers at one of the organizations for instance reported *"The Hackathon was a great success in terms of attendee numbers, but we see no surge in contributions in the months after the meeting. Was it worth the investment?"*

Acknowledgement

We want to thank the case study companies for their participation in this research. Furthermore, we thank Ryan Yang, Domen Smuc, and Lamia Soussi for their work on the literature and their writing comments. Finally, we thank all other members of the Software Ecosystems Laboratory at Utrecht University.

Bibliography

References

[1] J. Bosch, From software product lines to software ecosystems, in: Proceedings of the 13th international software product line conference, Carnegie Mellon University, 2009, pp. 111–119.

[2] K. Manikas, Revisiting software ecosystems research: A longitudinal literature study, Journal of Systems and Software 117 (2016) 84–103.

[3] O. Barbosa, C. Alves, A systematic mapping study on software ecosystems, in: Proceedings of the International Workshop on Software Ecosystems, 2011.

[4] M. Goeminne, T. Mens, A framework for analysing and visualising open source software ecosystems, in: Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE), ACM, 2010, pp. 42–47.

[5] M. Cusumano, Staying Power: Six Enduring Principles, Oxford University Press, 2012.

[6] K. Manikas, K. M. Hansen, Software ecosystems–a systematic literature review, Journal of Systems and Software 86 (5) (2013) 1294–1306.

[7] H. H. Olsson, J. Bosch, Ecosystem-driven software development: A case study on the emerging challenges in inter-organizational r&d, in: C. Lassenius, K. Smolander (Eds.), Software Business. Towards Continuous Value Delivery, Springer International Publishing, 2014, pp. 16–26.

[8] S. Jansen, M. A. Cusumano, S. Brinkkemper, Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry, Edward Elgar Publishing, 2013.

[9] S. Jansen, E. Bloemendal, Defining app stores: The role of curated marketplaces in software ecosystems, in: Software Business. From Physical Products to Software Services and Solutions, Springer, 2013, pp. 195–206.

[10] M. Iansiti, R. Levien, The keystone advantage: what the new dynamics of business ecosystems mean for strategy, innovation, and sustainability, Harvard Business Press, 2004.

[11] P. J. Williamson, A. De Meyer, Ecosystem Advantage: How to Successfully Harness the Power of Partners, California Management Review 55 (1) ((2012)) 24–46.

[12] C. Alves, J. Oliveira, S. Jansen, Software ecosystems governance-a systematic literature review and research agenda, in: Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS), Vol. 3, 2017, pp. 26–29.

[13] S. Jansen, R. Yang, The software ecosystem governance maturity model: Detailed practice descriptions, in: Submitted to Data in Brief, currently on www.softwareecosystems.org, 2020.

[14] T. de Bruin, R. Freeze, U. Kulkarni, M. Rosemann, Understanding the main phases of developing a maturity assessment model, Proceedings of the Australasian Conference on Information Systems 2005 (2005) 109.

[15] M. Van Steenbergen, R. Bos, S. Brinkkemper, I. Van De Weerd, W. Bekkers, The design of focus area maturity models, in: International Conference on Design Science Research in Information Systems, Springer, 2010, pp. 317–332.

[16] I. van den Berk, S. Jansen, L. Luinenburg, Software ecosystems: a software ecosystem strategy assessment model, in: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, ACM, 2010, pp. 127–134.

[17] A. Baars, S. Jansen, Proceedings of the Third International Conference on Software Business, ICSOB 2012, Cambridge, MA, USA, June 18-20, 2012., Springer, Berlin, Heidelberg, 2012, Ch. A Framework for Software Ecosystem Governance, pp. 168–180.

[18] J. van Angeren, C. Alves, S. Jansen, Can we ask you to collaborate? analyzing app developer relationships in commercial platform ecosystems, Journal of Systems and Software 113 (C) (2016) 430–445.

[19] J. Becker, R. Knackstedt, D.-W. I. J. Pöppelbuß, Developing maturity models for it management, Business & Information Systems Engineering 1 (3) (2009) 213–222.

[20] J. K. Crawford, Project management maturity model, Taylor & Francis, 2007.

[21] M. Paulk, Capability maturity model for software, Encyclopedia of Software Engineering.

[22] C.-E. Mols, N. Martin-Vivaldi, M. Werther, M. Ahlgren, K. Wnuk, Principles for Industrial Open Source, 2018.

[23] A. Arsanjani, K. Holley, The service integration maturity model: achieving flexibility in the transformation to SOA, in: Proceedings of the IEEE International Conference on Services Computing, IEEE Computer Society, 2006, p. 515.

[24] M. van Steenbergen, R. Bos, S. Brinkkemper, I. van de Weerd, W. Bekkers, Improving is functions step by step: The use of focus area maturity models, Scandinavian Journal of Information Systems 25 (2) (2013) 35–56.

[25] C. Wohlin, Guidelines for snowballing in systematic literature studies and a replication in software engineering, in: Proceedings of the 18th international conference on evaluation and assessment in software engineering, ACM, 2014, p. 38.

[26] R. Yin, Case study research: Design and methods, Vol. 5, Sage Publications, Incorporated, 2008.

[27] M. Schief, Business models in the software industry: the impact on firm and M&A performance, Springer Science & Business Media, 2013.

[28] H. van der Schuur, S. Jansen, S. Brinkkemper, The power of propagation: on the role of software operation knowledge within software ecosystems, in: Proceedings of the International Conference on Management of Emergent Digital EcoSystems, ACM, 2011, pp. 76–84.

[29] J. Bosch, P. Bosch-Sijtsema, From integration to composition: on the impact of software product lines, global development and ecosystems, Journal of Systems and Software 83 (1) (2010) pp. 67–76.

[30] M. Anvaari, S. Jansen, Evaluating architectural openness in mobile software platforms, in: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, ACM, 2010, pp. 85–92.

[31] S. Jansen, Opening the ecosystem flood gates: Architecture challenges of opening interfaces within a product portfolio, in: Proceedings of the European Conference on Software Architecture, 2015.

[32] J. Knodel, K. Manikas, Towards reference architectures as an enabler for software ecosystems, in: Proceedings of the 4th International Workshop on Software Ecosystem Architectures (WEA 2016), 2016.

[33] T. McDonnell, B. Ray, M. Kim, An empirical study of API stability and adoption in the Android ecosystem, in: Proceedings of the 29th IEEE International Conference on Software Maintenance, IEEE, 2013, pp. 70–79.

[34] E. Ververs, R. van Bommel, S. Jansen, Influences on developer participation in the debian software ecosystem, in: Proceedings of the International Conference on Management of Emergent Digital EcoSystems, ACM, 2011, pp. 89–93.

[35] S. Hess, J. Knodel, M. Naab, M. Trapp, Engineering roles for constructing ecosystems, in: Proceedings of the 4th International Workshop on Software Ecosystem Architectures, 2016.

[36] K. van Baarsen, S. Jansen, S. España, Measuring tool and resource maturity in developer ecosystems., in: Proceedings of the International Workshop on Software Ecosystems, 2017, pp. 88–102.

[37] P. Artz, I. van de Weerd, S. Brinkkemper, J. Fieggen, Productization: transforming from developing customer-specific software to product software, in: International Conference of Software Business, Springer, 2010, pp. 90–102.

[38] L. Soussi, Z. Spijkerman, S. Jansen, A case study of the health of an augmented reality software ecosystem: Vuforia, in: International Conference of Software Business, Springer, 2016, pp. 145–152.

[39] Z. Yang, S. Jansen, X. Gao, D. Zhang, On the future of solution composition in software ecosystems, in: Proceedings of the International Conference on the Economics of Grids, Clouds, Systems, and Services, Springer, 2016, pp. 3–18.

[40] T. Espinha, A. Zaidman, H.-G. Gross, Web API growing pains: Stories from client developers and their code, in: Proceedings of the Conference on Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), IEEE, 2014, pp. 84–93.

[41] E. den Hartigh, M. Tol, W. Visscher, The health measurement of a business ecosystem, in: Jansen, S., Cusumano, M., Brinkkemper, S. Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry. Edward Elgar Publishers, 2013.

[42] K. Wnuk, K. Manikas, P. Runeson, M. Lantz, O. Weijden, H. Munir, Evaluating the governance model of hardware-dependent software ecosystems–a case study of the axis ecosystem, in: Software Business. Towards Continuous Value Delivery, Springer, 2014, pp. 212–226.

[43] R. Basole, J. Karla, On the evolution of mobile platform ecosystem structure and strategy, Business & Information Systems Engineering 3 (5) (2011) pp. 313–322.

[44] T. Mettler, P. Rohner, Situational maturity models as instrumental artifacts for organizational design, in: Proceedings of the 4th international conference on design science research in information systems and technology, ACM, 2009, p. 22.

[45] T. C. Lacerda, C. G. von Wangenheim, Systematic literature review of usability capability/maturity models, Computer Standards & Interfaces 55 (2018) 95–105.

[46] E. Capra, A. I. Wasserman, A framework for evaluating managerial styles in open source projects, in: IFIP International Conference on Open Source Systems, Springer, 2008, pp. 1–14.

[47] S. Jansen, S. Brinkkemper, J. Souer, L. Luinenburg, Shades of gray: Opening up a software producing organization with the open software enterprise model, Journal of Systems and Software 85 (7) (2012) 1495–1510.