The Journal of Systems and Software 85 (2012) 1495–1510

Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Shades of gray: Opening up a software producing organization with the open software enterprise model

Slinger Jansen[a,*], Sjaak Brinkkemper[a], Jurriaan Souer[b], Lutzen Luinenburg[b]

[a] Utrecht University, P.O. Box 80.089, 3508TB Utrecht, The Netherlands
[b] GX Software, Wychenseweg 111, 6538 SW Nijmegen, The Netherlands

ARTICLE INFO

Article history:
Received 29 April 2010
Received in revised form 5 December 2011
Accepted 5 December 2011
Available online 19 December 2011

Keywords:
Software ecosystems
Product software vendors
Platforms
Software development governance
Organizational openness

ABSTRACT

Software producing organizations are frequently judged by others for being 'open' or 'closed', where a more 'closed' organization is seen as being detrimental to its software ecosystem. These qualifications can harm the reputation of these companies, for they are deemed to promote vendor lock-in, use closed data formats, and are seen as using intellectual property laws to harm others. These judgements, however, are frequently based on speculation and the need arises for a method to establish openness of an organization, such that decisions are no longer based on prejudices, but on an objective assessment of the practices of a software producing organization. In this article the open software enterprise model is presented that enables one to establish the degree of openness of a software producing organization. The model has been evaluated in five interviews, is illustrated using three case studies, and shows that organizational openness and transparency are complex variables, that should not be determined based on belief or prejudice. Furthermore, the model can be used by software producing organizations as a reference for further opening up their business, to stimulate the surrounding software ecosystem, and further their business goals.

1. Introduction

Software producing organizations (SPOs) have one goal in common: get the software that is being produced to be adopted as frequently as possible. Different business models exist to obtain that goal, from open source models where source code is given away for free, to service models where the source code is running on a protected server. SPOs have discovered that they can achieve their goals quicker by creating an ecosystem of partners around the product, such as developers or resellers. These developers and resellers will only participate, however, if the SPO opens up: business processes that are traditionally closed need to be shared with actors in the ecosystem to help them create value. Such value is created by mass-development (open source), mass-customization (open configuration), and many other strategies that include opening up business processes. Traditionally monolithic and closed organizations that open up their business processes are named Open Software Enterprises (OSEs).

An OSE creates its own network of collaborators called a software ecosystem. Software ecosystems are defined as a set of actors functioning as a unit and interacting with a shared market for software and services, together with the relationships among them (Jansen et al., 2009). These relationships are frequently underpinned by a common technological platform or market and operate through the exchange of information, resources and artifacts. There are several motivators for establishing a software ecosystem, such as the ability to respond more flexibly to varying requirements for varying domain specific solutions that are based on the platform or quicker adoption than competitors through large partner and developer networks. Iansiti and Levien (2004) state that organizations in an ecosystem depend on each other mutually for survival and effectiveness. Therefore, the health of individual organizations is not solely affected by itself but heavily depends on the health of the complete network of firms.

An OSE is defined as an organization that produces software and that has opened up its processes to at least one of the actor types that are present in its software ecosystem. OSEs can be any type of SPO, from independent software vendors to open source organizations. An actor can be any participant in a software ecosystem. In this article we identify four types of actors, being developers, value-added-resellers (VARs), service partners, and customers. Each of these actors has the capability to add value to a SPO, by adding

* Corresponding author. Tel.: +31 619884880.
E-mail addresses: slinger.jansen@uu.nl (S. Jansen), S.Brinkkemper@uu.nl (S. Brinkkemper), jurriaan.souer@gxsoftware.com (J. Souer), Lutzen.Luinenburg@gxsoftware.com (L. Luinenburg).

0164-1212/$ – see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2011.12.007

source code, by making sales for the organization, or simply by paying the software organization directly.

When a SPO intends to become an OSE, it must review its business model. Several different business model evaluation techniques exist, such as the application of Jansen and Brinkkemper's software ecosystem modelling methods (Boucharas et al., 2009) or the evaluation technique of Rajala et al. (2003) who define a software business model along four axes, being the distribution model, the services and implementation supplied by the software vendor, the revenue logic it applies, and its product strategy. In this article is shown that the business model plays a key part in establishing how a SPO opens up its processes and that different value adding actors, such as third-party developers and VARs, have interest in the opening of different process domains.

Several national initiatives, such as the draft 'policy on Open Standards for eGovernance' (Ministry of Information and Communication Technology, 2008) in India and the 'UK Government IT strategy' (Chief Information Officer Council, 2010), have called local governments to use 'open' software and standards instead of closed alternatives. It is hard for these public organizations to establish exactly how open a SPO is, especially because at present no objective instruments exist. This is detrimental to the software industry, since commonly SPOs are unjustly deemed to be open or closed, leading to poor purchasing decisions.

The model presented in this article aims to establish how open or closed an SPO is. The model's raison d'etre is found in three aspects. First, unjust purchasing decisions are made based on prejudiced openness evaluations, thereby not letting the best product or supplier win. Secondly, SPOs are unaware of the different openness options that exist, and they are unaware of how they can safely change their business model and open up their business to comply with the requirements of local government and to stimulate the software ecosystem. Thirdly, the model tries to take the focus away from "open source", and attempts to show that there is more to openness than an open software license.

This article continues with the research approach in Section 2, including how design research was performed to create the OSE model and how case study research was applied to evaluate the model. In Section 3 the OSE model is presented that lists all the different openness options a SPO has. Furthermore, an openness measure is presented that assists in objectively assessing the degree of openness of a SPO. In Sections 4, 5, and 6, the results from three case studies are presented, including how their degree of openness has been established. The case results are further analyzed in Section 7 and discussed in Section 8, followed by related work in Section 9. The article is concluded in Section 10 with a summary of the contributions.

2. Research approach

The main result from this research constitutes the OSE model. The OSE model has been developed using design science, applying the method as specified by Alan Hevner et al. (2004). Design-Science research has a set of seven guidelines which help the researcher conduct, evaluate and present design-science research. The seven guidelines address *problem relevance*, *design as an artifact*, *research rigor*, *design as a search process*, *design evaluation*, *research contributions* and *research communication*. The guidelines of the research problem, designing as an artifact, research rigor, and design evaluation are discussed.

The *research problem* that was identified is that the term "openness" is used frequently in the context of SPOs, usually as a synonym for beneficial, without actually providing a definition of openness in this context. Unfounded decisions are made to open up parts of SPOs and unsupported procurement decisions are made based on prejudiced perception of openness in the SPO's context. These unfounded decisions can lead to wrong strategic decisions and can potentially make good products less appealing, based on a speculative qualification towards openness. Two *sub-problems* were formulated: (1) there currently is no model that provides managers an overview of the openness measures that can be taken by an SPO and (2) there is no way to measure how open an organization is. The results of these problems are that software organizations need to guess in which way they should open up their processes to leverage the software ecosystems and that software organizations are judged prematurely for being too closed. Both results are detrimental to the software industry. The *artifact that is developed* is the OSE model, which addresses the two problems identified. The OSE model can be used as a reference framework by SPOs to establish which options to open up if an SPO decides to become more ecosystem friendly. Also, the model can be used by procuring organizations to establish which of the shortlisted products are more "open", by establishing which of the openness options have been chosen by the supplier of those products.

In regards to *research rigor*, the research has been done using two cycles of the design cycle, where a different evaluation method was used for each cycle. For the first cycle the artifact was evaluated doing five interviews with experts in the domain of software ecosystems. In the second cycle, three case studies were done to further establish whether the model was useful and could be applied to examples of OSEs. Both research cycles were performed by a different researcher.

2.1. Evaluation interviews

Five interviews were organized with experts in software ecosystems. These experts were two product managers, a CTO and a CEO of a software company, and finally an ecosystem manager. Each of the interviewed has had at least 10 years of experience in the software industry. The product managers were both from GX Software, one of the case study companies. Interviews generally lasted approximately 90 min and consisted of 10 open ended questions about openness (as shown in Table 1), a model evaluation, and a final case identification. The open ended questions concerned the definition of an OSE, the different ways in which the interviewee perceived openness, and how some companies appeared to be more open than others. The questions are categorized into problem relevance, model feasibility, and model evaluation. These questions were established using a thought map and the guidelines for design research as presented by Hevner et al. (2004) who explicitly state relevance, feasibility, and evaluation for artifact (in our case the OSE model) development. The answers to the questions were noted down during the interviews, and processed and stored in a spreadsheet within two hours of the interview.

After discussing the 10 questions about openness, the model evaluation started with putting the graphic representation of the model on the table and discussing each item in the model in detail. Many potential additions were written down during these interviews. (15 change suggestions in total, of which 6 were implemented in the model.) An example of an openness option that did not make the model, is "platform evangelism", since, though relevant to an 'open' business model, it does not specifically concern opening up software development or its governance (Bonaccorsi and Rossi, 2003). In the last two interviews, the OSE model was deemed complete by the interviewees and no further changes were suggested to the OSE model.

2.2. Case studies

Three case studies were selected by going through a list of approximately 200 university contacts. Four companies were

Table 1
Introductory questions for the expert reviews (before the model was shown).

Question	Category
How do you perceive organizational openness in the context of SPOs?	Problem relevance
What do you believe are the main reasons for opening up a SPO?	Problem relevance
Do you feel that SPOs are sometimes wrongfully judged to be open or closed?	Problem relevance
Do you believe a model could be created that measures the openness of a SPO? What would it look like?	Model feasibility
What companies do you believe are open? Which ones closed? What has convinced you of the fact that they are open or closed?	Model feasibility
Do you think that there are domain specific aspects to openness of SPOs? Can a producer of banking software even be open or is that a business model related choice?	Model feasibility
Do you believe that the openness initiatives, like the NOiV in the Netherlands, or similar programs in India, the UK, and other countries, can actually stimulate openness in an economy?	Model evaluation
Do you think that a more open SPO is stimulating its ecosystem in a better fashion than a closed organization?	Model evaluation
Do you believe that SPOs have enough information to open up their businesses? Do you think an overview with openness options is necessary?	Model evaluation
Do you believe that being too open may threaten the business model of a SPO?	Model evaluation

approached for an evaluation, and two were willing to participate, being GX Software and the Open Design Alliance (Jansen et al., 2000). The third case study of the Eclipse Foundation was added to complement a closed source case and a community source case with a case that provides its source code as open source completely. The case studies followed the guidelines of Yin (2003), and Jansen and Brinkkemper (2008). According to these guidelines a case study report and a case study database were created. It must be noted that the EF case study did not include any interviews, but was done through document study. The case study results were presented to the case study participants to see whether they agreed that the openness options correctly represented reality in both the openness options themselves and the overall openness overview. The case study participants and results are described further in Sections 4, 5 and 6.

The case studies consisted of three steps. First, a generic description of the case was created, that described the organization's ownership, structure, software product(s), and some of its recent history. The second step consisted of checking the openness options of the OSE model with the interviewees, to establish which of the options were considered to be open. The third step consisted of discussing how and why some of the openness options were actually open or closed, and how these openness options related to the business model of the organization. These three steps can be found in the structure of the case study reports, as well as the short summaries in this paper.

The model was applied to the case studies by asking, for each openness option, how the organization rated itself in regards to openness. For each option where the interviewee considered the organization open or closed, we asked the interviewee to give the motivating reasons behind saying this option was open or closed, such that in a discussion a qualitative rating could be established. All assignments were, however, done by the researcher. The reports written after the case study were sent back to the case companies for evaluation. In one case this led to extended discussions and some change suggestions for the model, which were not included.

3. Open software enterprise model

Three motivators are identified for opening up a software business. The fundamental goal behind each of these motivators is the increased adoption of a software artifact: as the barriers (i.e., price) to use the product are lowered, customers will sooner use the product. The first motivator is idealism. Once a software business is opened up fully, the organization inexplicitly states that it believes in its products and their adoption. With these open aspects, more actors in the ecosystem are able to exert their influence over the development of the products and keep it alive. A second motivator is survival. If an organization is competing with another one, opening up might encourage quicker adoption and standardization of its products, thereby outperforming the non-free competitor. Examples of these are IBM's Eclipse and Mozilla's Netscape, which were both opened up to compete with the de facto (commercial) solution. The third motivator is usually a commercial one: openness can, for example, provide customers with (part of) the functionality for free, but when they want to use the software commercially, they need to upgrade to a commercial license.

The open software enterprise model presented in this article is defined as listing the different *openness options* a SPO has, to open up (parts of) the organizational governance, the software development processes, the software product management processes, the marketing and sales processes, and the consultancy and support processes. These openness options can have both opening and closing effects on the business model. Each of the openness options concerns a decision to share something that would not be shared by a 'traditional' closed SPO. One of the compelling cases for the OSE model lies in the integration of supply chain partners, such as VARs and external software developers. The concept of the OSE and its supply chain integration is based on the concept of the Extended Enterprise, as presented by Davis and Spekman (2003). The OSE model is based on the premise that openness is not a matter of fully open or fully closed: organizations can choose to have a *degree of openness*.

The OSE model is displayed in Fig. 1. The OSE model has been created along two dimensions, being the SPO Practices Dimension and the Management Level Dimension. Across the Management Level Dimension, there are three layers being strategic, tactical and operational (from long-range-view to short-range-view). As a rule of thumb, each lower openness management level concerns a shorter managerial view: whereas strategic management generally concerns 5–10 years from now, operational management is concerned with the running of an organization. These three openness management levels (strategic, tactical, and operational) are taken from traditional management literature, such as the books by Peter Drucker (1954). Along the SPO Practices Dimension, five openness domains, governance, software development, product management, marketing and sales, and consulting support and services, are based on Brinkkemper's research model on product software (Xu and Brinkkemper, 2007) and Riehle's five core business processes of SPOs (Riehle, 2009), although Riehle has renamed governance to "community management". The research and development openness options in the model were identified in large part by going through the CMM-i (CMMI Product Team, 2006). The software product management parts were validated by comparing and contrasting with van de Weerd's product management framework (van de Weerd et al., 2006).

A keystone player is an actor in the ecosystem, whose contribution to the ecosystem stimulates the health of the entire ecosystem. A specific type of keystone player is the technology provider, which supplies the platform, software, and/or standards that are being used by a large set of actors in the ecosystem. A keystone player is not defined by its size, but by its contribution to the overall health of the ecosystem. The OSE model is directed at actors in the software

	Governance	Research and Development	Software Product Management	Marketing and Sales	Consulting and Support Services
Strategic	- Open up governance - Create partnership model - Open up IP strategy - Coordinate contributions to other ecosystems - Share competition policy - Share acquisition strategy - Implement ecosystem knowledge mgmt strategy	- Share technology and research roadmap - Share development process knowledge - Stimulate open standards - Share source code - Apply for joint research and development funding	- Share product lifecycle plans for products - Share platform strategy and vision	- Share market vision - Develop innovative business models - Create sales partner program	- Share services delivery management strategy
Tactical	- Enforce development process standard - Provide partners with governance procedures - Coordinate grievances - Help partners in IP conflicts	- Share innovations - Support interchangeable data formats - Share source code policy - Create reuse policy - Outsource tasks - Certify third-party components	- Outsource requirements engineering to partners - Share and adjust product (line) roadmap(s) - Manage third party IP in product	- Share market information - Share customer and supplier information - Develop distribution channels	- Outsource implementation projects to partners - Share ticket database - Share project process knowledge - Develop and share quality measures
Operational	- Make ecosystem explicit - Create a partner directory - Create user groups - Use and create reusable software licenses	- Create and publish (content) APIs and SDKs - Create reuse enabling architecture - Open up testing process - Share bug repository - Do co-development - Provide developer training - Propagate software operation knowledge	- Open up requirements management process - Open up release planning process - Share release candidates	- Certify partners - Create internal and external component markets - Involve partners in marketing and sales	- Share implementation knowledge - Share (customer) configuration knowledge - Use collaborative workspaces for customer communication - Provide consultant training

Fig. 1. The open software enterprise model.

ecosystem that are directly in contact with the keystone technology provider, also known as partners who are in direct connection to the keystone firm in the software supply network (Farbey and Finkelstein, 1999). Partners are defined as enterprises functioning as associates in an activity, endeavor, or sphere of common interest. When opening up, customers and partners surrounding the software organization are pulled into the organization and may become active participants in previously closed processes. When looking at the software ecosystem perspectives (being actor, software supply network, and software ecosystem level) (Jansen et al., 2009), the OSE model is focused on the software supply network level. It must be noted that there exist critical requirements for an open organization to succeed. The most important requirement is that the number of partners and customers that are able and willing to participate in the processes of an open organization is large enough.

3.1. OSE: governance

Governance is defined as the way an organization is managed, including its powers, responsibilities and decision-making processes (Dubinsky and Kruchten, 2009). Generally, an organization's governance is set out in its constitution or legal identity. Governance of software development involves the assignment of roles and decision rights, as well as the measures and policies that enable continuous assessment. In the context of this article, it also defines how much power is left to the community and how much the software developing organization 'keeps for itself', since coordinating software development in a supply chain is different from developing within one organizational entity (Riehle, 2009).

On the **strategic level (governance)**, an organization can decide to fully open up its governance policies (in a read or even write fashion), thereby giving the community around the organization insight or even power in the organization. An example of opening up governance policies is appointing an open committee that decides what new releases of a product will contain. A precondition is that the ecosystem surrounding the organization has been made explicit and some kind of partnership or membership model is in place. A second part of opening up governance policy, is the opening up of the Intellectual Property (IP) that is created by the organization, where IP does not only refer to the source code, but also to any other artifacts developed within the organization, up to process descriptions and product lifecycle plans. The strategic level of opening up governance is related strongly to dealing with competition and other ecosystems. To begin with, the organization can decide to contribute or even integrate into other ecosystems. Furthermore, ecosystems can have explicit or implicit strategies on how to deal with other (competing) ecosystems. Furthermore, an organization can choose to explicitly share its acquisition policy, i.e., they way in which the organization selects businesses to acquire and patents to acquire. Finally, an organization must create a knowledge management strategy, to enable and empower partners in the software ecosystem, without surrendering too much information and value. An example can be the opening up of a bug tracking system, which informs the community of what bugs are still open, but also informs competitors of the weaknesses in the product.

On the **tactical level (governance)** an organization can open up by helping partners in a number of ways: first it can provide or even enforce a development process on its partners. Secondly, generic governance processes can be shared and propagated to partners developing components or plug-ins for the platform that is provided by the organization. Thirdly, the organization can help partners in solving grievances among each other and even assist them in IP battles.

On the **operational level (governance)** an organization has three options to become a more open organization: the organization can make the ecosystem explicit, by stating what types of partners are part of the ecosystem and how these are related. Furthermore, the organization can create a partner directory, in which

partners and customers can find (other) partners and customers. Also, the creation of active user groups can help these customers in combining their forces and requests, and thereby providing better feedback cooperatively to the SPO. Finally, an organization can choose to create licenses that can be reused by other actors in the ecosystem.

3.2. OSE: Research and Development (R&D)

Research and development (R&D) spending in a software business on average consists of 25% of total revenue (Rönkkö et al., 2009). The R&D department plays the most important part in opening up the software business, since most valuable knowledge is created there. Also, R&D is aware of potential partners: at some point during development these partners raised their heads with information requests, API questions, or even requests for source code. Because of this, typically the first requests for openness come from the R&D department.

At the **strategic level (R&D)**, the R&D department can decide to share the technology roadmap, concerning the upcoming challenges that are relevant to the domain on which the software business focuses. Immediately following the technology vision comes the research vision, concerning the challenges that the R&D department is going to focus on in the following years. There are several organizations that have become proficient at presenting these visions, such as Apple with their closely followed press events and Microsoft at its worldwide DevDays. Another option R&D departments have is to publish the development procedures, with different aims. One aim can be to provide transparency, but more frequently, these procedures are published to inspire ecosystem participants to follow these same procedures. Another strategic option for an R&D department is to support open standards, as opposed to closed standards. These open standards turn out to be healthy for an ecosystem (Bannerman and Zhu, 2008), since proprietary standards tend to be accompanied by several expensive licensed products. Examples of such closed standards are, for instance, the DWG format (opendesign, 2011) from AutoCAD and the PSD Photoshop image format from Adobe.

One of the most defining options is to share the actual source code of a product. Opening up the source code can be done at different levels. These levels range from making the source code readable by partners in a consortium to writable by anyone. The degree of openness completely depends on the business model that is followed. If partners pay 'rent' to be part of the ecosystem, the first might be favorable. If the providing organization only sells services, the source code might just as well be writable by anyone willing to improve the product. An option software businesses also have is to apply for research funding to further stimulate open standards and open knowledge. The vendor can apply for research funding because its technology does not only further the software business goals, but also that of its surrounding partners.

At the **tactical level (R&D)**, the R&D department can choose to share innovations that are not directly related to the core value of the product. If the organization is not planning to further research the viability and business model behind an innovation, it might be worthwhile to share the innovation with partners who are interested in that particular domain. Another option is to support interchangeable data formats that enable data sharing between different products (i.e., different ecosystems.)

A further option is to share the source code and reuse policies. These source code policies prescribe how one should reuse other source code (from the web, for instance) and how source code shall be published. Also, a reuse policy defines which licenses are allowed to be considered for inclusion into the main product. Unfortunately, it still occurs frequently that software businesses discover that third-party components that are used within some of the products have an incompatible license, as Alspaugh, Asuncion, and Scacchi observe (Alspaugh et al., 2008). Another option on the tactical level is the outsourcing of tasks, to achieve scalability. As these tasks are outsourced, other partners are included in the production process, bringing them even closer to the production process. A vendor that has been successful at this in the past is Cisco (Gawer and Cusumano, 2002). Finally, an organization can certify components of other actors in the ecosystem to provide them with a quality rating, which these actors can use for sales and promotion.

On the **operational level (R&D)**, a software business can open up several operational parts of the development process, such as the bug tracking system and the testing process. Also, in regards to the software architecture, several steps can be taken to open up the software product. First and foremost, the products can be opened up using (content) Application Program Interfaces (APIs), for partners to enable functionality and content reuse. Secondly, as the product grows, it might be relevant to modularize and create a reuse enabling architecture. Such an architecture stimulates reuse both within the R&D department and for others who wish to reuse parts of the product.

On the operational side, R&D can open up the testing process, by sharing both test results and candidate releases of the product. This transparency enables dependent parties to pre-test and potentially adjust their products appropriately. Also, the sharing of the bug repository can be beneficial to third parties in that this provides insight into potentially unexpected behavior of the product. Furthermore, doing co-development with others on the product and providing developer training to third-parties can be beneficial to the ecosystem. Finally, software operation knowledge, i.e., the knowledge that can be gathered from software operating in the field such as performance data and crash reports, can be propagated to ecosystem actors to further stimulate higher quality software in the ecosystem (van der Schuur et al., 2011).

3.3. OSE: Software Product Management (SPM)

Software product management (SPM) is defined by Ebert et al. as the process of managing software that is built and implemented as a product, taking into account lifecycle considerations. It is the discipline and business process which governs a product from its inception to the market or customer delivery and service in order to generate biggest possible value to the organization (Gorschek et al., 2010). SPM is different from governance in that it focuses more on the internal specifics of a product, such as its requirements, quality, development plans, and market plans, whereas governance focuses more on organizational and intellectual property ownership, development processes, organizational strategy, and community management.

On the **strategic level (SPM)** organizations can choose to share both their platform strategy and vision, by presenting it at yearly meetings. Furthermore, the SPM level managers can choose to publish their process and life-cycles in regards to products, so that others gain insight and potentially reuse the lifecycle for their own practices. By doing so, others can hook into similar development practices, creating a more homogeneous ecosystem.

On the **tactical level (SPM)** organizations can choose to outsource requirements engineering to others, such that other parties engage in the gathering of requirements for future versions. Furthermore, organizations can opt to open up the product roadmaps. These can be opened up such that others can see where the product is headed and base their own decisions on these road maps. The product roadmaps can also be opened up in the sense that others can adjust the roadmap, giving up more power. A typical way of doing so is by organizing customer days in which customers can provide their opinions and inputs to the product, for instance

through a voting process (Kabbedijk et al., 2009). Finally, the organization must make explicit and share the policies in regards to the IP of a product, and the IP of the products included in the product itself (such as open source products).

On the **operational level (SPM)** organizations can choose to open up the requirements engineering process, for instance by letting customers vote on new proposed features. Furthermore, organizations can open up their release planning, where they coordinate with stakeholders when a next release is published. This coordination is generally beneficial for stakeholders, because then the stakeholders can adjust their planning to that of the organization. Also, an organization can publish release candidates, such that stakeholders in the ecosystem can test with the release candidate before a new version comes out.

3.4. OSE Marketing and Sales (M&S)

On the **strategic level (M&S)** an organization can open up the market vision and expectations, to encourage partners to pursue certain domains and discourage them from pursuing others. Also, the organization can choose to develop innovative new business models for its partners. An example of such an innovative business model is the in-application payment model on the iPhone, which makes it possible for application developers for the iPhone platform to make money through the application itself and not just by sales of the application. These innovative business models open up new opportunities for partners. Another major option is to develop a progressive partner model, in which the most valuable and high-profile partners het control in the organization as well.

On the **tactical level (M&S)** an organization can choose to open up and share its market information with the aim of stimulating the competitive edge of the full ecosystem. Another information source that stimulates the ecosystem is the publication of (parts of) current and potential customers and suppliers within the ecosystem. Finally, new distribution channels can be developed by an organization to enable partners to sell software through channels that were previously unavailable. An example of new distribution channels is application stores, such as Android's market, in which developers can sell applications globally to all Android users.

On the **operational level (M&S)** the organization can choose to create internal and external component markets, to encourage the use of these channels for sales of applications and domain specific solutions. Furthermore, the organization can encourage partner organizations by handing out certifications based on quality criteria. These certifications will inform customers of these partners of the level of quality they can expect from these partners. This can happen in regards to sales and services as well as component certification. Finally, an organization can involve partners in selling software. An example is when a customer visits a partner of the organization, to get a reference on the organization. Another example is by providing partners with incentives for selling the main platform and its components.

3.5. OSE Consulting and Support Services (CSS)

Almost every SPO will at some point provide Consulting and Support Services (CSS) (Cusumano, 2008). These services are required to help customers get set up, help partners in deploying the product for the first couple of times, and help plug-in developers in getting their first version up and running. As these services are provided, the organization benefits from making explicit and sharing knowledge with partners, since the knowledge does not need to be made explicit when the organization provides these services again. In closed ecosystems, keystone players often restrict the transfer of implementation knowledge using policies or procedures in order to secure core competencies (Simonin, 1999). As shown in the ERP implementation context, the transfer of knowledge between technical consultants and business users contributes to the success of implementation (Ko et al., 2005).

On the **strategic level (CSS)** the organization can share all knowledge that is gathered while providing CSS to partners. The knowledge that is gathered by the organization and its partners can benefit future CSS projects, and storing these centrally and opening them up to partners benefits the ecosystem. Furthermore, the development of an explicit strategy for these processes, such as delegating smaller projects to partners, benefits the software ecosystem in that partners will get new business opportunities from the organization.

On the **tactical level (CSS)** an organization can delegate CSS projects to partners. The ecosystem profits from this in two ways: the partner can sell CSS and the partner learns new things about the platform provided by the organization, creating valued skills. Furthermore, on the tactical level CSS knowledge must be shared using project process knowledge and ticket databases, preferably in which partners can help each other and share knowledge among each other as well. Finally, an organization can decide to develop quality measures for customers and partners, to indicate at what service quality level the partners are operating, and what they can do to improve these levels.

Finally, on the **operational level (CSS)** organizations can share detailed descriptions of CSS project details. One such detail can be the sharing of customer configuration, though this will be uncommon. Furthermore, an organization can employ collaborative workspaces in which different project organizations work together and share knowledge on a platform. Also, organizations can provide consultant trainings to accredit consultants and assure customers.

A business model focuses on how value is added to an organization. Four types of external entities that can add value to a SPO are distinguished. An organization opens up its enterprise depending on which of these external entities an organization wishes to gain value from. The four value adding entities distinguished for this article are developers, customers, VARs, and service partners. Developers are a special case, since their value is added through source code, instead of money.

If the software is developed in an open source environment, governance issues require much more attention than a closed development organization. The governance issues that deserve attention are the development process, the role of software developers, and the owner of the IP. Also, the research and development domain of openness is influenced: the technology and research roadmaps and vision must be shared explicitly with the community, the IP must be explicitly managed and checked in the source code, and bug repositories and test reports must be opened up. Finally, software product management is affected since now it is required that all developers can have their say in the release planning and requirements that will be implemented in the future.

Three different distribution channels can be identified, indirect-through-VAR, indirect-through-service-organization, and direct-to-customer (Popp, 2010). It must be noted that a combination of the first two can be possible, but that is only marginally relevant for the openness options. The first distribution channel, indirect-through-VAR, requires openness on the research and development domain, especially when looking at the APIs and SDKs required to extend a platform. Also, depending on how much of the income these resellers provide, openness is required in both the marketing and sales and consulting and support domains. The same holds for the second distribution channel, indirect-through-service-organization, which requires openness in both the marketing and sales and consulting and support domains. The third distribution channel, direct-to-customer, requires openness on almost all domains, since the customer has a large stake in

the future of the SPO. Its focus, however, is on buying and then deploying and implementing the software solution in the customer organization.

3.6. Model application

The model is to be used by two different parties, being external and internal assessors. Internal assessors, who have access to all stakeholders in the organization, assess the organization's openness with the aim of providing insight to the organization into the available and implemented openness options of an organization. An external assessor is someone who needs to know how open an organization is in comparison to another, for instance for purchasing decisions. These external assessors may not be able to uncover the details for each of the openness options, but a fair comparison can still be made of organizations when they are assessed on their visible characteristics.

Due to the explorative nature of this work, it is hard to provide a strict method for the determination of whether an option is open for an SPO or not. We do, however, wish to point out the following guidelines. To begin with, the option's openness must count for the product or platform that is responsible for the main portion of the company's income, revenue, and activity (in the case of an organization without revenue or easily-measurable income). Secondly, the options are all based on the availability of some critical concept, such as the availability of another channel than the main distribution channel for "Develop distribution channels". This critical concept is usually visible through either the SPO's web site but can also be established by querying partners. Thirdly, there must be critical mass behind the openness option: i.e., it cannot be a the simple "boasting" of an organization, but other players must be able to participate and collaborate with the SPO. This collaboration must be observable and preferably measurable. Finally, openness suggests that any entity is able to participate and collaborate in the ecosystem. In case some entities are by default excluded from the collaboration despite proven attempts to collaborate, the option is simply not open.

The model is applied by assessing for each openness option whether it is implemented by an SPO. For each of the openness options, one must qualitatively assess whether it is open or not. Some are easy to establish, such as the option "Share technology and research roadmap", since that can be found through a web search or through contact with the company. If the roadmap is only shared with a restricted set of third parties, such as a partner network, it is not considered open. In the words of one of the experts: *"The Apple iPhone platform is open in a sense that you can develop for it, but by their rules, so actually it is not that open."* Others, such as "Develop distribution channels" are harder to establish, since it may seem unclear what qualifies as a distribution channel. For the case studies "Develop distribution channels" was considered open, when an app store was present or when partners were involved in the distribution of the main software product. The criteria for being open or closed were established based on the descriptions provided in this section. In the case of the case studies, it was always established that we were talking about the main product of the SPO, and not a product that does not necessarily represent the strategy of the company. For example, the release of Microsoft's Wix deployment tool set as open source does not make Microsoft a company with the "share open source" openness option.

3.7. Model relationships

The model is not easily separable, i.e., most of the options have direct effects on the other options. We have been able to identify several types of relationships in the model, based on a matrix of all dependencies between openness options. The matrix has not been presented here for three reasons: it is hard to operationalize the dependencies, the matrix stands unevaluated at this time, and it does not directly add to the clarity of the OSE model. Some examples and categories of dependencies, however, are described below. The first type of relationship found is horizontal top to bottom relationships, where one openness option from a management level depends on an openness option from a higher management level. This category contains the most relationships. An example of such a relationship is the operational option "Create internal and external component markets" which is influenced by the tactical option "Develop distribution channels", which in turn is influenced by the strategic option "Develop innovative business models". These relationships are abundant, illustrating the strong internal cohesion of the model.

The second type of relationship we name diagonal relationships. These diagonal relationships, which tend to go from top left to bottom right, are relationships that show that strategic openness options do not only influence their own SPO practices, but also others. A trivial example of such a relationship is "Involve partners in marketing and sales", which is directly influenced by "Create a partnership model". Interestingly, some of the options, especially in the governance SPO practice, influence many other options. The options that have 10 influences or more are "Create a partnership model", "Implement ecosystem knowledge management strategy", "Share source code" and "Share technology and research roadmap".

Some of the other relationships identified are in the same practice category at the same management level (within one block in Fig. 1). Furthermore, there are no influences from lower management levels to higher management levels. There are some influences from practice to practice at the same management level, such as "Share development process knowledge" from the research and development practice, being influenced by "Implement ecosystem knowledge management strategy" from the Governance practice. Finally, all of the practices at the two lower management levels are influenced by or related to options from higher levels.

4. Case study: The Open Design Alliance (ODA)

The Open Design Alliance (ODA) is an organization that strives for open standards with respect to CAD formats, and more specifically the *dwg* file format. The ODA presently has 2000 members. The ODA develops the ODA platform, which includes the ODA libraries. The ODA Platform can be included in any CAD product to enable reading and writing of *dwg* formatted files. Currently approximately 28 people work for the ODA, of which 25 are developers. The ODA has four different member types, being associate, commercial, sustaining, founding and educational. Each of these membership types has different rights, privileges, and costs, up to the level of full access to the source code. Membership is gained after signing the ODA membership agreement. The ODA maintains its ecosystem by organizing conferences, publishing books and learning materials, and a bi-weekly status report that is sent by e-mail to each member.

4.1. ODA software ecosystem developments and health

Three software ecosystems play a part in the ODA's history. First, the firm's software ecosystem consists of all those who use the ODA's components and platform. Secondly, the ODA is a major player in the *dwg* technology ecosystem. The ODA has an interesting history in the context of software ecosystem s. First, the ODA was founded as a company by eight other companies around 1990, feeling the need to open up the *dwg* file format. In 1998 the Visio corporation, now part of the Microsoft Corporation, acquired the company. Soon the Visio corporation realised that the expertise

within their newly acquired team was not specific to Visio, but to many others as well. Visio founded the OpenDWG Alliance, or ODA as it is now known, in 1998 and opened up its membership to others.

Another interesting development is that the ODA started out as an organization that focused on opening the *dwg* standard to software vendors. During its lifetime, however, the focus widened to include several other complementing and competing standards into the libraries. As the amount and variety of formats that were included in the libraries increased, the structure of the software evolved as well. Presently, the ODA platform consists of viewers, readers, APIs and documentation that enable developers to access and modify CAD related storage formats. Along with these developments came a name change: the ODA libraries became the ODA platform.

Many of the current utilities in the ODA platform were supplied or purchased from third-parties, which thereby enriched the ODA platform. As these third-party utilities were included in the platform with increasing frequency the ODA realised that the platform needed to encourage this type of reuse. Recently, the ODA started the *Third-party Supplier Program*, a program that enables sustaining and founding members to contribute their own components to the platform. The ODA platform is then used as a reseller platform, although the license agreements and billing are arranged by the members themselves. Finally, the ODA platform is trying to be more effective for its members by focusing on vertical markets. Presently, the ODA platform contains components for the geo-information market, the mechanical design market, and for the architecture market. The ODA is researching other areas that are of interest to its members.

The openness of the ODA software ecosystem is a threat to the business model of the *dwg* format of AutoDesk, which has led to a number of lawsuits from AutoDesk towards the ODA. These lawsuits are a relatively successful competition strategy, considering that they tend to cost the ODA a lot of money, a resource that is normally used to employ platform developers. The ODA software ecosystem can be considered to be in direct competition with the AutoDesk software ecosystem, since AutoDesk does provide APIs to access the *dwg* format to its own members.

The ODA measures its success in the number of members it has. The members are the most valuable asset to the ODA, since they enable, through their membership fee, the ODA to get more developers. The ODA actively brings together its members to create a more active community, since the members are dispersed all over the globe.

4.2. ODA and its degree of openness

The ODA is a special kind of organization: due to the fact that it consists of 25 developers and a small number of managers, it has no sales department and no consultancy and services department. The ODA has, however, decided to open up many of the processes that are present in the OSE model in Fig. 1.

In regards to **ODA governance**, the ODA is led by its founding members. These founding members can do anything, from changing the ODA's licenses to completely open source to stopping the organization completely. On a day-to-day basis, however, the founding members tend not to exert their control and put their trust in the ODA CEO. Because the ODA is not a big organization, it has specifically made choices about which processes to open and which processes to leave for the future. To begin with, the governance of the ODA is rarely changed but open for changes by the founding members and change requests from the other members. The ODA has an advanced partnership model, with different memberships. At the basic level, new members pay approximately 100 dollars for a one time trial membership. At the top level, members pay 12,000 dollars yearly, for access to premium services and the source code of the complete platform. The ODA does not do any acquisition of partners and competitors, at present. The ODA does not enforce its partners to use specific development standards and procedures. The ODA also does not coordinate grievances between partners. At present, the ODA ecosystem has not been made explicit and no member directories exist, even for members themselves. The ODA makes explicit to its members how trademarks and IP should be used within the ODA ecosystem.

In regards to **ODA Research and Development**, the ODA publishes its technology and research roadmap, though be it informally at the yearly conference and through its newsletter. At present, the ODA does not make its development process explicit to members. The ODA does explicitly address the issue of open standards, mostly through the constant discussion with AutoDesk to open up its DWG file format. The ODA does not, however, apply for funding, mostly because it is a global organization. Source code is openly available to the top level memberships, although it is not frequently asked by these members. The ODA at present has no reuse policy and does not outsource development tasks. ODA members communicate using the ODA forum, which is frequently studied by ODA management, to discover problems, trends, and challenges for the ODA community.

The **ODA's Software Product Management** is fairly closed, although the roadmap is approved by the committee of founding members. The plans for new products are published, but not the lifecycle of these products. The platform strategy and vision are shared informally, much like the technology and research visions. The ODA does not share or outsource requirements engineering and product roadmaps. Furthermore, the release plans are not shared explicitly with members. Recently, a wish list has been made available online, where partners can enter new product requirements.

ODA Marketing and Sales is the smallest portion of the ODA's activities. It is therefore not surprising that the ODA does not have an explicit partner model for sales and implementations, does not develop innovative business models, share market information, or develop distribution channels. Furthermore, the ODA does not certify partners or involve partners in sales. The ODA has recently introduced the concept of sharing components on a component market, although this has not been operational to date.

ODA Consulting and Support Services consist mostly of online help through the forum. Furthermore, personal support (also generally through the web) is given to higher level members of the ODA. The ODA shares its knowledge on how to integrate the ODA platform mostly through its documentation and forums. Implementation projects are not outsourced to partners. The ODA at present does not certify its partners and there are no quality measures in place by which partners can qualify themselves. Also, no information on end-customers is shared, because the ODA is not a direct contact for the customers of its members. Finally, the ODA does not provide developer trainings.

4.3. ODA Business Model Alignment

The ODA has two main activities, being governance and software development. In regards to governance, the ODA has about five people who work on keeping the ecosystem active and alive. These five are involved in management, administration, product management, governance, and marketing. The other 25–30 employees work on the ODA platform, the main product of the ODA. The ODA has an open governance model and has the ambition to be as open an ecosystem as for instance the Eclipse ecosystem. The ODA is restricted in size, however, limiting its abilities to open up certain processes, such as sharing product life cycles or sharing the product line roadmaps. Because the ODA's main source of income is resellers, it is expected that the ODA has opened up mostly the

domains of R&D, SPM, and Marketing and Sales, which is true. CSS is also expected to be more open in regards to CSS, but because the platform can be deployed relatively easily, not much is required from the ODA in that respect.

The ODA is defending the case for open standards: it reverse engineers the dwg format of commercial Autodesk solution, to open up an otherwise expensive proprietary standard. There are dangers to this business model, since AutoDesk consciously and actively protects the IP of its formats. AutoDesk has taken action against the ODA, by disputing, for instance, the dwg trademark (a case that was settled out of court). It remains a question whether the current shape the ODA is in (small overhead) can remain if competing ecosystems continue to take action against the ODA. On the other hand the consortium structure has made the ODA relatively resilient to influences from other ecosystems.

The ODA is a rare case of a SPO that is not allowed to make profit: every dollar made should go back into further development of the platform. This has several implications for the software ecosystem. Because the ODA is run by members, a small layer of management (four people) is required to steer the organization. All other personnel are approximately 25 developers who add value to the platform. An interesting effect of this structure is that the ODA has a high level of credibility with its members, since the members feel they can influence decisions that are made by the ODA, and not too much money is wasted on overhead, such as high management fees or large marketing campaigns. The ODA has experienced continuous growth over the last years, even though 2009 was a tough year for the software industry. Its consortium structure and the popularity of the dwg format for three-dimensional drawings have been increasing the demand for the ODA platform. More members have been actively participating in the governance of the organization, providing the developers with much needed feedback and new requirements.

5. Case study: the Eclipse Foundation

The Eclipse Foundation (eclipse.org, 2011) is a non-profit, member-supported corporation that hosts the Eclipse projects, a range of products that support developers in the development process through compilers, development environments, visualization tools, etc. The Foundation provides governance services, such as running the IT infrastructure, doing IP due diligence, and mentoring the open source projects, for the Eclipse community. The EF can be seen as one of the leading open source communities. It is generously supported by industry.

IBM, who was then developing the first versions of Eclipse, released it into open source in 2001. In that same year the Eclipse Consortium was started by nine companies who felt that current development tools were not sufficient and should not be developed by one organization alone. By 2003 the consortium had grown to over 80 members and in 2004, the EF was founded.

Much like the Open Design Alliance described in Section 4, the history of the EF provides a jumping board for a successful ecosystem. When the Eclipse platform was released into open source, members of the consortium were free to develop their own domain-specific solutions for Eclipse, immediately creating a lively community around the platform. Now, only nine years after the first consortium was created, the EF and the developers surrounding the ecosystem consist of tens of thousands of developers, fourteen strategic members, and several other rungs of members, such as solution, committer, enterprise and associate members. Interestingly, strategic members and enterprise members pay fees to the EF from 50,000 dollars up to 500,000 dollars. These fees above 50,000 dollars can be reduced to that amount by committing developers to the Eclipse projects full-time.

5.1. Eclipse and its degree of openness

The EF adheres to most of the options of the OSE model. To begin with, in regards to **EF Governance** the EF has fully opened its governance strategy. The members of the EF can, depending on their level of membership and overall influence in the ecosystem, change EF policies according to the EF bylaws and membership agreement. Furthermore, the EF has created an advanced membership model, accommodating the different types of members (users, developers, VARs). Also, the EF has an explicit IP strategy that is fully explained on its site. The EF also coordinates contributions to other ecosystems, such as the Apache ecosystem, as decided by the members. The EF does not share an explicit acquisition strategy, although several open source projects have joined with the EF. Also, no explicit strategy has been created for how to deal with competition of the EF. There is a well-defined knowledge management strategy for the EF, resulting in several portals, forums, and documents. One part of this are the development process standards that are described in full on the EF website and can be adopted by both EF projects and outside projects. The EF can mediate when members need arbitration in a conflict within a project or between projects. Finally, the EF has created user groups explicitly with an associated membership type, the ecosystem has been made explicit, and there exists a large partner directory on the EF website.

For **EF Research and Development**, the EF works actively on promoting its technology and strategy roadmaps. Furthermore, the EF communicates openly how its software is developed. The EF further stimulates open standards, by explicitly opening up the different file formats and by reusing open standards where possible. Also, the EFs source code is accessible through its CVS portal (dev.eclipse.org, 2011). The EF shares its IP, including any innovations that do not fit the EF's strategy. Furthermore, the EF has made explicit what kind of code can and cannot be reused in projects. The EF does not certify any of the available third party plug-ins, although a certification component for Eclipse (Sherriff and Williams, 2006) is available. The Eclipse project has created an easily extendible architecture, that tries to promote reuse across the board. Eclipse project unit test results are published, though be it informally. Furthermore, the Eclipse project's bug repository is open (bugs.eclipse, 2011). Finally, Eclipse lets its members help development, developer training is provided, and reusable software licenses have been created.

In regards to **EF Software Product Management**, the EF shares the product lifecycle for different products (or projects). Furthermore, the EF explicitly shares the EF platform strategy and vision. The EF 'outsources' requirements engineering to members, shares and adjusts product roadmaps based on member opinions, and the requirements process of the Eclipse project is completely transparent. Finally, release planning is made explicit, and release candidates are usually published ahead of time, to accommodate members that depend on the Eclipse Product for their own future.

With respect to **EF Marketing and Sales**, the EF shares its market vision and develops innovative business models, in the sense that platform plug-ins can be built of different shapes and sizes, with different business models, as long as the EF membership agreement is respected. There is no sales partner program for the Eclipse project, but the EF does share whatever marketing information it can find on the Eclipse platform, as well as potential members, partners, and developers. The EF does not certify partners. The EF does, however, include partners in promotional activities. Also, the EF has created several component markets, that can be used by third party developers to promote their plug-ins and products.

In regards to **EF Consulting and Support Services**, the EF has no explicit implementation projects, where the Eclipse Project needs to be deployed, configured, and implemented at a customer. The

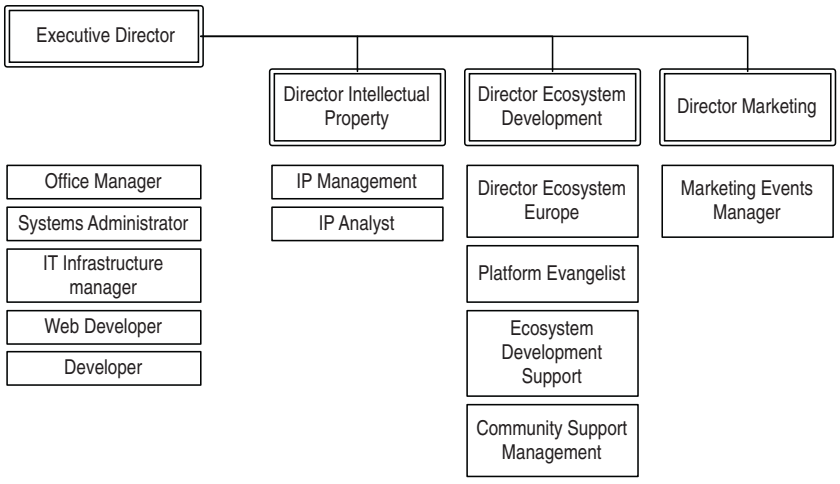

Fig. 2. Eclipse Foundation Management Structure, recorded on April 16th, 2010. Note the absence of a development department.

EF therefore does not have a service delivery management strategy and does not outsource any such project. The EF does not have a specific ticket database for implementation problems, nor does it share implementation project process knowledge or quality measures. Finally, the EF does not provide consultant training.

5.2. EF Business Model Alignment

When looking at the business model of the EF, we see that developers add significant value to the organization by their daily contributions to the IP of the EF. Furthermore, a large portion of revenue comes from customers and end-users (enterprise members) because of their dependency on the Eclipse platform. Also, a second portion of revenue comes from VARs, who use the platform to create new value for their customers. A final note in regards to the Eclipse platform is its mode of distribution: Eclipse can be downloaded from the EF website and deployed fairly easily. Taking the design of the EF business model into account, the degree of openness of the EF is easily explained. The openness in the governance, R&D, and SPM domains is necessary to accommodate the developers and VARs. The openness in the marketing and sales domain is required by VARs. Finally, the fact that there is hardly any openness in the area of CSS lies in the fact that little or no services are required for the deployment and implementation of Eclipse into an organization.

The EF case is different from the previous two cases because of the EF's main activity. The EF's main activity is the governance of the Eclipse ecosystem. The Eclipse team is modeled in the management structure in Fig. 2. The main functions that can be distinguished are intellectual property management, ecosystem development, and marketing, which is a totally different structure than that of, for instance, the ODA case, where 30 out of 34 people are involved with software development and quality assurance.

The Eclipse ecosystem is successful for a number of reasons. To begin with, the EF has been well able to mobilize its users, domain specific developers, and 'verticals', i.e., the third parties that develop plug-ins for a specific development technology, such as an Eclipse environment for PHP (PDT) or Ruby on Rails (Aptana). A second reason for being successful is that Eclipse is a platform for developers, by developers. Developers and software engineering researchers predictably require more features from a platform such as the Eclipse platform and these developers are well suited to develop new extensions and features. This is different from the ODA case, where the end users rarely know they are using a component from the ODA when they do. A third reason why the EF is successful is the strong backing from industry. The founding members of the EF are among the most respected SPOs in the world.

6. Case study: GX Software

GX Software is a Dutch company with approximately 120 employees, active in both the Netherlands and the United States. GX Software's product, GX WebManager is a CMS that is used by many large organizations to build and maintain complex websites and web applications. The architecture of GX Software WM has been studied and modeled by Souer and van Mierloo (2008). GX Software has approximately 150 customers, with several product deployments each. GX Software's revenue comes for approximately 50% from licenses and approximately 50% from professional services, such as partner enablement, implementation projects, customization, and training. GX Webmanager, GX Software's product is extendible, in that customizations can be built and managed for specific customers or verticals. GX WebManager is dependent on several components supplied by the Apache Foundation and GX Software contributes actively by frequently participating in open source development of Apache components.

GX Software's business model has changed considerably in the last decade, as can be see in Fig. 3. Traditionally, GX Software consisted of one core team of developers and one core team of service engineers that provided customer specific deployments (*2000*). These two groups cooperated intensively, leading to complex management structures: sometimes features were requested for all customers, sometimes for one customer specifically. As both the company and its development department grew,

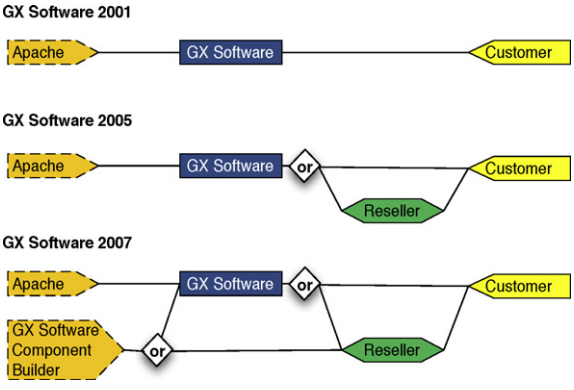

Fig. 3. Business Model Evolution for the GX Software Company.

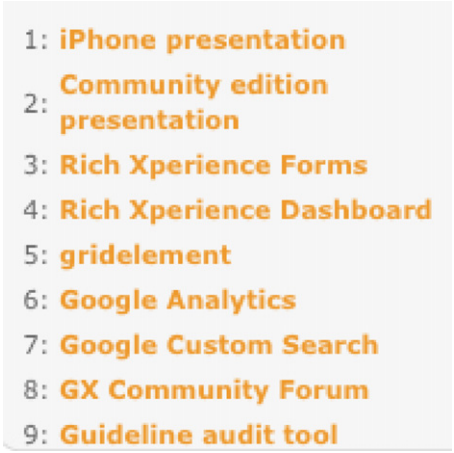

Fig. 4. Screenshot from GX Software AppStore: GX Software's component store. The number one application renders pages from any deployment of GX Software Web Manager for the iPhone.

a functional separation was made between product developers and project developers. Also, requests from service engineers were now handled by project developers first, to make sure that no customer-specific requirements were developed by the core platform team.

Fig. 4

The company's management team decided in 2005 to change their strategic course. They knew that the current way of development and implementations would not by scalable with ambitions to grow both domestic and international. As the customer base neared its 100th customer, in an economy where developers and service engineers are scarce, GX Software intensified their partnering model, expecting it to be more successful and scalable in the long run, where partners would do customer specific implementations and GX Software would focus on its platform: GX WebManager.

GX Software currently has a strong focus on the verticals Publishers and Media and Finance (Banks and Insurance). However, over the years they have built a diverse customer base: different vertical industries require different custom solutions. Furthermore, domain specific solutions are required, which are so large in numbers that it would be challenging for one company to develop and maintain each plug-in, API, and interface connector. Also, partners were increasingly interested in more advanced APIs and development kits to further build extensions on top of the current platform. Therefore, from 2007 onwards, the GX Software's platform was rebuilt, which enabled partners to develop their custom components, and GX Software introduced an online component exchange marketplace, where components can be published under different licenses (open, free-closed, and commercial-closed). Currently, this marketplace accommodates over 1000 components on the GX Software AppStore. One of the more successful measures GX Software has taken is build a component certification tool that automatically verifies and certifies third-party plug-ins. The certification tool enables GX Software to quickly see whether the component does not break any interaction standards and that it lives up to GX Software's quality criteria.

6.1. GX Software and its degree of openness

GX Software has traditionally been a services company with its proprietary software product as a foundation for their services, in that it has always provided a large amount of services next to its license sales. Furthermore, the founders, all academics, have an open view on the software business and are willing to experiment with new business models. It is therefore not surprising that GX Software already at an early stage (2007) started the development of a component store. Furthermore, GX Software has actively contributed to several open source Apache projects, indicating that GX Software is willing to give back to the community. The business model has always been, however, to sell both licenses and services.

In regards to **GX Software governance**, GX Software does not have an open governance or IP strategy, does not share its acquisition strategy, and does not share its policy with partners on how to deal with competition. GX Software has developed a simple partnership model. GX Software does not tell third-party developers how to deal with competition or whether or not to deal with other ecosystems. GX Software has made its ecosystem explicit and has an explicit knowledge management strategy in regards to sharing development and implementation knowledge with partners. To be more specific, GX Software has created different web portals, such as the GX Software AppStore.com portal for sharing and selling components and a GX Software developer website for developers to share knowledge. Also, GX Software has a partner directory that lists partners for implementation and development activities surrounding the GX Software product.

In regards to **GX Software Research and Development**, source code of example components is shared through the WCMExchange and product architects write blogs on specific topics, usually with source code included. The core software product however is closed source. The road map and new innovations (new product development) are shared with a select group of industry experts, certified partners, and customers. Also, new innovations that are not essential to the product are shared. Examples of this are components that are placed in the WCMExchange by GX Software developers. The expert group provides feedback and gets access to release candidates of the software. GX Software shares, in the form of academic papers and a development blog and guidelines, how they develop software. GX Software also shares the technological and research visions through presentations and blogs. GX Software has an internal policy in regards to source code reuse, since it reuses several open source components, and shares them with certified partners and customers. GX Software has created an SDK for developers to build new applications against and GX Software certifies, using an automated tool, these third party components. The certification tool has been well received (and used) by these third party developers. GX Software also applies a reuse enabling architecture, although components are mostly used within GX Software's main product. At present, the testing process is completely closed. The bug repository is open to third party developers. GX Software does share release candidates and have them tested by pilot customers and partners. Furthermore, GX Software presently does some co-development. It does do all kinds of trainings, such as user, developer, and implementation trainings. Finally, GX Software applies several open standards in its products.

For **GX Software Product Management**, the organization shares explicitly its platform vision through blogs and presentations. Furthermore, product roadmaps and release plans are published regularly for both partners and customers. These are based on company vision, market trends and generalization of the customers and users. However, the roadmap is not adjusted to the specific wishes of individual customers. The requirements management process is steered in part by requirements from partners and customers that are provided through a portal. GX Software does not outsource the requirements engineering process to partners.

GX Software Marketing and Sales at GX Software consists mostly of stimulating partners to sell the GX Software product, for which a partner network has been created. Domain specific projects are distributed to specific partners who are specialized in that domain. GX Software has also created new business models for its partners with its component store.

In regards to **GX Software Consulting and Support Services**, GX Software has made quick advancements in supporting its partners.

To begin with, GX Software distributes incoming projects among its partners. Secondly, in regards to implementation knowledge, a ticket database is available to all partners where implementation specific issues are discussed and solved. Thirdly, GX Software has developed several quality measures which are discussed informally with each partner. Also, GX Software provides trainings for consultants and developers of third parties. Finally, GX Software has a professional services (PS) organization to support partners during the implementation process if requested.

6.2. GX Software Business Model Alignment

GX Software's main activity consists of developing software and assisting VARs and service partners in providing successful implementation projects for customers. GX Software's product can be downloaded as a community edition, but is more commonly distributed through partners and resellers, which accounts for the main sources of income. GX Software presently does not depend on external developers to supply code, so in regards to governance everything is relatively closed. In regards to R&D GX Software is still discovering the plethora of openness options, but because GX Softwares ecosystem is relatively young it has not taken the opportunity to fully open up the R&D and SPM domains. On the domains of M&S and CSS GX Software is open in most respects, as predicted by the OSE model.

The application domain is suitable for the implementation of a software ecosystem strategy (Bosch, 2009): the market in which GX Software is active requires both vertical solutions (governments, industries, etc.) and horizontal solutions (plug-ins, viewers, content import and export features, etc.) that cannot be built by one organization alone in an efficient way. However, productivity of third parties is still a challenge in the GX Software ecosystem. GX Software suffers from the bootstrapping problem (Jansen et al., 2009) because there are not enough participants in the ecosystem to gain critical mass quickly. Recently, however, a quick increase has been detected in the number of components in its component store, leading us to believe that the GX Software ecosystem strategy is working.

7. Analysis

In this section the findings extracted from the case studies are presented. One of the main contributions of this article lies in the fact that the **business model and strategy determine the degree of openness** of a SPO. In Table 3 is speculated that domains that are most open are related to the types of actors (third party developers, resellers, service partners, and customers) that the organization wishes to stimulate. For example, it can be seen that when a SPO wishes to have developers add value to the organization, it must open governance, research and development, and software product management. Product distribution determines whether service and implementation knowledge is required. In the example of SAP, one of the world's largest enterprise resource planning SPOs, for instance, a wide network of partners exists that perform implementation of their complex configurable products. On the other hand, for a product such as Eclipse, which generally works out-of-the-box, less openness is required in the area of deployment and servicing.

The EF case makes us hypothesize that **in community open source development, governance plays the largest part**. If software governance were not in the hands of the members of the EF themselves, these members would not be willing to contribute their money or their time to the Eclipse ecosystem. Furthermore, the fact that the Eclipse platform is released as open source, encourages developers of new components to also open up their components, creating a lively ecosystem to which developers gladly contribute, due to the profit they make from that ecosystem.

The ODA case shows that **in a closed consortium, development is the main activity.** The consortium is expecting value for their money, and this value should lie in the main product of the consortium (software, in the case of the ODA). Governance is in the hands of small number of members, instead of any party in an open source consortium, so these members expect to get the most out of the consortium. This results in a low overhead organization, such as the ODA.

When studying the histories of the three organizations, it is observed that **software ecosystems are created by opening up**. In the case of GX Software, it started externalizing the deployment, configuration, and customization of its main product. In the case of the ODA, the governance to the development of an essential component for many CAD software developers was opened up, to enable them to control the further development of the ecosystem. In the case of Eclipse, the source code was opened up to maintain a critical mass of developers and progressively gain influence over more expensive commercial alternatives. One of the more revealing observations is that some of the more commercial organizations, such as for instance Apple and Microsoft, though generally considered highly closed organizations, have had to open up large parts of their M&S and CSS domains. It is not so much true that these organizations are 'closed' per sé, but they have tried to maintain as much control as possible by not opening up their governance model. It is thus not the 'closedness' that is detrimental to the software industry, but their level of control.

Although openness of an organization tends to be seen as a good thing for a software organization, **openness is not always beneficial to the organization.** In the example of GX Software, much effort has been spent in opening up the organization. It was hypothesized that giving up parts of the IP, such as consultants' knowledge, would increase the critical mass of the GX Software platform, thereby increasing turnover. However, even though this knowledge was opened up, GX Software did not start making more money immediately. It seemed that the critical mass of the software ecosystem first needed to grow. This growth period to critical mass introduces risks, which must be considered by organizations trying to follow the same strategy.

The ecosystem manager is responsible for keeping the different actors in the ecosystem satisfied and for developing new policies and strategies based on the organization's strategy towards openness. The role of the ecosystem manager is relevant on all domains of the SPO, as the OSE model illustrates. **The role of partnership and ecosystem manager is rising in SPOs**. In all three objects of study, the role and its responsibilities have been made explicit. The partnership or ecosystem manager is generally part of the R&D or M&S department.

Two other findings are further explained below being the facts that **openness creates 'new' business models** and that **licenses do not determine membership alone**.

7.1. Elaboration on Openness and New Business Models

An interesting question is what motivates an organization to open up the software development process. These decisions are made with the vision that it will grow or preserve the success of the functionality that is created by the SPO. It does appear, however, that the more open an organization becomes, the more radically the business model needs to be adjusted (Riehle, 2009). Some interesting examples of new business models can be found abundantly in industry. As one of the experts mentioned *"We see some new developments in the industry, such as hybrid business models with professional licenses coexisting with open source licenses for the same code. It is no longer just about the (compiled) code itself, there's the*

Table 2
Assessment of Openness in SPOs (GX = GX Software).

Domain	S/T/O	Openness option	ODA	ECL	GX
Governance	S	Open up governance strategy	X	X	
	S	Create partnership model	X	X	X
	S	Open up IP strategy	X	X	
	S	Coordinate contributions to other ecosystems		X	
	S	Share competition policy	X		
	S	Share acquisition strategy			
	S	Implement ecosystem knowledge management strategy	X	X	X
	T	Enforce development process standard		X	
	T	Provide partners with governance procedures		X	
	T	Coordinate grievances	X	X	
	T	Help partners in IP conflicts	X		
	O	Make ecosystem explicit	X	X	X
	O	Create a partner directory		X	X
	O	Create user groups		X	X
	O	Use and create reusable software licenses		X	
Research and development	S	Share technology and research roadmap	X	X	X
	S	Share development process knowledge		X	X
	S	Stimulate open standards	X	X	X
	S	Share source code	X	X	
	S	Apply for joint research and development funding		X	
	T	Share innovations		X	X
	T	Support interchangeable data formats	X	X	X
	T	Share source code policy	X	X	X
	T	Create reuse policy	X	X	X
	T	Outsource tasks			
	T	Certify third-party components			X
	O	Create and publish (content) APIs and SDKs	X	X	X
	O	Create reuse enabling architecture	X	X	X
	O	Open up testing process		X	
	O	Share bug repository	X	X	X
	O	Do co-development		X	X
	O	Provide developer training		X	X
	O	Propagate software operation knowledge			
Software product management	S	Share product lifecycle plans for products		X	
	S	Share platform strategy and vision	X	X	X
	T	Outsource requirements engineering to partners	X	X	
	T	Share and adjust product (line) roadmap(s)		X	X
	T	Manage intellectual property of and in product	X	X	X
	O	Open-up requirements management process	X	X	X
	O	Open-up release planning process		X	X
	O	Share release candidates		X	
Marketing and sales	S	Share market vision	X	X	X
	S	Develop innovative business models	X	X	X
	S	Create sales partner program	X		X
	T	Share market information		X	
	T	Share customer and supplier information		X	
	T	Develop distribution channels	X		X
	O	Certify partners			X
	O	Create internal and external component markets	X	X	X
	O	Involve partners in marketing and sales		X	X
Consulting and Support Services	S	Share services delivery management strategy			
	T	Outsource implementation projects to partners			X
	T	Share ticket database			X
	T	Share project process knowledge	X		X
	T	Develop and share quality measures			X
	O	Share implementation knowledge			X
	O	Share (customer) configuration knowledge			
	O	Use collaborative workspaces for customer communication			
	O	Provide consultant training			X

business model around it as well." Here we highlight two short examples of RedHat Linux and MySql.

Red Hat is a publicly listed company that has been successful in making profit from open source code, and is most famous for the Red Hat Linux operating system. Red Hat makes profit by selling services, instead of licenses, such as subscription-based customer support, trainings, and software quality assurance, for (partly) open software products. Interestingly, Red Hat is using different licenses for different products. Some products are completely open source, whereas other products (and even versions) have more closed licenses. These more closed licenses generally state that the source code is only available to subscribers of certain support services. The company is constantly experimenting with openness and making adjustments to the business model. Red Hat is an example of a company that still largely relies on third parties from the open source community (although increasingly on its own customers as well) to supply new components and products for niche problems that Red Hat could never solve independently. Red Hat serves as a prime example of a SPO that stems from the open source world, but makes profit successfully in a closed industry.

Table 3
Business model and value adding actors.

	Governance	R&D	SPM	M&S	CSS
Developers	x	x	x		
VARs	x	x	x	x	x
Service orgs				x	x
Customers	x	x	x		x

Another example is provided by the product MySQL, that can provide the database layer in any software product. MySQL, recently acquired by Oracle, is sold using *multi-licensing*, i.e., the same source code is distributed with different licenses under different conditions. An open source product, for instance, can use MySQL freely, whereas a commercial closed source product can only reuse and resell MySQL under a commercial license. The concept where multiple licenses are used for different groups of customers is also known as market segregation. Part of MySQL's income comes from services, much like in the Red Hat example.

7.2. Licenses do not determine membership alone

Table 4 lists the different actor types that have been identified explicitly within the three cases. GX Software, for instance, distinguishes four different types of actors in this context, being service partners, VARs, component builders, and customers. Customers, VARs, and service partners add value to GX Software by paying for licenses. Developers add value to GX Software by building new components for the GX Software platform. One of the contributions of this article lies in the identification of these actor types.

Another interesting observation is that different members of an ecosystem, such as EF enterprise and strategic members, pay different amounts of money for the same rights to the source code (van Angeren et al., 2011). It must be noted, however, that the difference in membership lies in other incentives than just the licenses: an enterprise member simply pays dues, whereas a strategic member can choose to pay the even larger dues by supplying Eclipse developer hours instead of money. Furthermore, strategic members are placed on a special strategic members web page, among which are companies such as SAP and IBM.

8. Discussion

The OSE model can play the role of a reference framework in the establishment of the degree of openness of a SPO. Some reservations must be made in regards to the different levels of openness per openness option, the scoping of the model, and the drivers of opening up a SPO. Furthermore, some threats to validity must be made explicit.

When trying to establish whether an organization is open for specific options, it was sometimes difficult to draw the line between purely open and closed. In the case of the ODA, for instance, source code is only available to the alliance members, whereas the source code is fully open with regards to the EF case. A similar example where it was hard to draw the line between open and closed was encountered trying to establish whether GX Software with its contribution to one open source Apache component constitutes that GX Software "coordinates contributions to other ecosystems". More research is required towards this degree of openness of specific options, and their effects on the business model. We consider it a future challenge to increase the fidelity of our instrument, where the different openness options are not valued across a boolean scale, but a continuous one. Furthermore, we believe that some of the openness options weigh heavier than others and that there may be relationships among different openness options. With a higher resolution the instrument will provide more precise results, but more data is required before the instrument can be adjusted like that. One suggestion is to include a generic onion model to show exactly how far certain openness options reach and base the openness degree on this reach as well.

The research method that has been followed is design research. The artifact that has resulted from the design cycles is the OSE model. Two design cycles have been completed, where in the first design cycle the model was verified by software ecosystem experts in interviews and in the second design cycle the model was verified by performing the three case studies described in this paper. One of the largest changes made to the model was the addition of the governance domain, after the discussions with experts. Multiple rounds of evaluation are required to further validate the model and establish that it is in its final state. The case study research guidelines of Runeson and Höst (2009) were applied to assure valid results from the case studies.

Strategic management of a SPO is a complex task that requires a long breath, vision, and entrepreneurial courage. When it comes to openness, this is clearly the case. The immediate benefits of opening up the business can only be found by experiment and measuring the actual benefits after opening certain options is a challenge. In the end, the health of the ecosystem determines whether it is ready for further openness and the resulting growth path. The research area of software ecosystems would be helped by further assessment methods (den Hartigh et al., 2006) that establish the maturity and readiness for openness within an ecosystem.

The research approach suited the maturity of the research topic well: explorative case studies can help define new phenomena and in that sense the data set for this research has been rich enough to create the OSE model. Although the OSE model was not changed much after the interviews, the model may not be complete. A future challenge will be to create a model with a higher fidelity, by studying the relationships between openness options and by defining different scales per openness option.

Table 2 provides an overview of the openness assessment at each of the case studies. Unsurprisingly, the EF has the highest degree of openness. It is surprising to see, however, that GX Software, a commercial organization, appears to be more open than the ODA, which is governed by a consortium. One of the main contributing domains to the openness degree is the CSS domain, which is hardly an issue for the ODA. When comparing organizations, however, all domains must be considered when evaluating the openness of a software enterprise.

A discussion point is whether these findings could be summarized into some kind of score or degree. One could imagine the introduction of a Software Producing Openness Degree (SPOOD). Such a SPOOD could consists of five components, being the five domains in the OSE model: governance, R&D, SPM, M&S, and CSS, where a weighted OSE score is calculated for each of the components. For each of the domains $D = \{g, r, s, m, c\}$ an openness score could be calculated for a company, where each of the domains could add equally to the final openness score, i.e., $|O_g| = |O_r| = |O_s| = |O_m| = |O_c| = 20$. However, to create and validate the SPOOD, more data is needed to create a model with a higher fidelity.

9. Related work

Software ecosystems and openness receive a lot of attention. The topic of software ecosystems lies at the crossroads of many different topics related to interaction between software products and companies. These interactions are for instance found when products reuse other products and require specific licenses (Alspaugh et al., 2008). Interaction is also found when looking at component reuse in open development communities, and the software ecosystem view has been used in that context as well (dos Santos and

Table 4
Ecosystem actor identification and licenses.

Organization	Identified value adders	Actor type	Dues	Use	Reselling	Developer	Access source
GX	Service partners	Service partners	high		x		
GX	Value added resellers	VARs	high		x	x	
GX	Component builders	Developers	None	x		x	
GX	Customers	Customers	high	x			
EF	Associate members	Customers	very low	x			x
EF	Solutions members	VARs	low	x	x		x
EF	Enterprise members	Customers	high	x			x
EF	Strategic members	Customers	very high	x			x
EF	Committer members	Developers	None	x			x
ODA	Educational members	VARs, customers	very low			x	
ODA	Associate members	Customers	low	x		x	
ODA	Commercial members	VARs	medium	x	x	x	
ODA	Sustaining members	VARs	high	x	x	x	
ODA	Founding members	VARs	very high	x	x	x	x

Werner, 2010; Le Berre and Rapicault, 2009). A specific view in the software ecosystem domain looks at the role of a keystone, i.e., a player that is of fundamental importance to an ecosystem. The works of Iansiti and Levien (2004), Bosch (2009), and den Hartigh et al. (2006) specifically look at this role. A development in the area of software ecosystems is the modeling of ecosystems with the aim of understanding the dynamics within such an ecosystem, which is for example done by Bala Iyer and Lee (2006). The work performed for this article incited us to perform more research in the area of partnership agreements, to establish how these affect the openness of an organization and how these partnership agreements influence the business model (van Angeren et al., 2011). For an extensive study of the ecosystem literature the systematic mapping study of Barbosa and Alves is useful (Barbosa and Alves, 2011).

With regard to openness, different works play a part in promoting open software and open organizations, with different agendas. In our work on the *clopenness* concept (te Molder et al., 2011), we further emphasize the gray area between open and closed, and define a model that assesses openness on a continuous scale. The fundaments of this model lie in availability, accessibility, transparency, reciprocity, and licensing, which each need to be evaluated to establish whether an option is open or not. The model is applied in the context of a software company. It is different from this work in that it attempts to further uncover the characteristics of each openness option and in that it incorporates the domain specificity of the product. Some of the questions asked by Messerschmitt and Szyperski (2003) have directly inspired some of the openness options in our model, such as for instance "certify partners" and "share market information". Another assessment that we have performed into openness, is that of mobile operating system platform architectures, where we find that the business model has a strong influence on the openness options in a software architecture (Anvaari and Jansen, 2010).

In regards to the promotion of open source, a lot of work is being done in the area of open communities (Raymond, 1999) and open source business models (Bonaccorsi and Rossi, 2003; Riehle, 2009; Scacchi, 2007). Also relevant is the work on openness of interfaces (in software ecosystems) such as the work by Cataldo and Herbsleb on open APIs (Cataldo and Herbsleb, 2010). Openness in the context of software ecosystems has not been made explicit and this article attempts to fill that niche. However, some literature is available on open standards and their role in software ecosystems (Bannerman and Zhu, 2008; West, 2007).

10. Conclusions

The results of our research indicate that the OSE model described in this article provides a complete overview of the different options and domains that can be opened up by a SPO, taking into account that the model was evaluated by five experts only. The model has been evaluated (with completeness as one of the criteria) through interviews with these experts and applied to three case studies. It has been established that the OSE model provides insight into openness options and can help SPOs in determining their openness strategies. The OSE model also shows that an organization can choose to be open on both the supply and demand side of the supply chain. Examples of this are the opening up development on the side of software developers and contributors, or opening up service delivery on the side of service partners who deploy, configure, and service the software platform produced by the organization. Furthermore, it is shown that the degree of openness is fully determined by the business model as developed by the decision makers of the SPO.

Other findings from this research are that openness can quickly create a critical mass of developers or partners around a product, if and only if the surrounding partners are ready and prepared to enter the ecosystem and take up one of the four roles: developer, VAR, service partner, or customer. Finally, a prerequisite for a vibrant ecosystem is an open platform.

This article proposes the OSE model to determine the degree of openness of a SPO. Presently, organizations are perceived to be either open or closed, frequently stigmatizing those considered 'closed' as being too commercial and unfavorable compared to open organizations. In this article it is shown that the decision to be open or closed is multi-faceted and cannot be judged without extensive study. With the further elaboration of the *openness scale*, software acquirers can honestly and explicitly measure the openness of the candidate products, thereby giving them an instrument (the model) to objectively assess the degree of openness of an organization. On the other side, a health model that establishes whether the community surrounding a platform or product is ready for moving towards an ecosystem is beneficial to SPOs: it would enable them to establish when to open and which options. The further elaboration of the *openness scale* and *software ecosystem health model* are considered future work. Furthermore, an interesting question is the finding of constellations among data sets from different organizations. We expect patterns to arise in these constellations based on characteristics such as non-profit organizations, (hardware) platform oriented organizations, multi-product organizations, and international organizations.

Acknowledgements

We would like to thank the case study participants for their openness and honesty in the interviews and case studies. Also, the comments from the anonymous reviewers have enriched the article significantly.

References

Alspaugh, T.A., Asuncion, H.U., Scacchi, W., 2008. The role of software licenses in open architecture ecosystems. In: Proceedings of the First International Workshop on Software Ecosystems, Washington, Virginia.

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

Bala Iyer, N.V., Lee, C.-H., 2006. Managing in a "small world ecosystem": lessons from the software sector, Harvard Business Review (5).

Bannerman, P., Zhu, L., 2008. Standardization as a business ecosystem enabler. In: Proceedings of the International Workshop on Enabling Service Business Ecosystems (ESBE'08), Sydney, Australia.

Barbosa, O., Alves, C., 2011. A systematic mapping study on software ecosystems. In: Proceedings of the third Workshop on Software Ecosystems, CEUR-WS, vol. 746, pp. 15–26.

Bonaccorsi, A., Rossi, C., 2003. Why open source software can succeed. Research Policy 32(7), 1243–1258 (Open Source Software Development).

Bosch, J., 2009. From software product lines to software ecosystems. In: Proceedings of the 13th International Conference on Software Product Lines, San Francisco, USA, pp. 111–119.

Boucharas, V., Jansen, S., Brinkkemper, S.,2009. Formalizing software ecosystem modeling. In: IWOCE '09: Proceedings of the 1st international workshop on Open component ecosystems. ACM, New York, NY, USA, pp. 41–50.

https://bugs.eclipse.org/bugs/.

Cataldo, M., Herbsleb, J.D., 2010. Architecting in software ecosystems: interface translucence as an enabler for scalable collaboration. In: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, ECSA '10, ACM, New York, NY, USA, pp. 65–72.

Chief Information Officer Council, United Kingdom, Government ICT Strategy: Smarter, Cheaper, Greener, 2010.

CMMI Product Team, 2006. Capability Maturity Model for Development, Version 1.2.

Cusumano, M.A., 2008. The changing software business: moving from products to services. IEEE Computing 41 (1), 20–27.

Davis, E., Spekman, R., 2003. Extended Enterprise, the Gaining Competitive Advantage Through Collaborative Supply Chains. FT Press.

den Hartigh, E., Tol, M., Visscher, W., 2006. The health measurement of a business ecosystem. In: Proceedings of the European Chaos/Complexity in Organisations Network (ECCON) Conference.

http://dev.eclipse.org/viewcvs/.

dos Santos, R.P., Werner, C.M.L., 2010. Revisiting the concept of components in software engineering from a software ecosystem perspective. In: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, ECSA '10, ACM, New York, NY, USA, pp. 135–142.

Drucker, P., 1954. The Practice of Management. Harper and Row, New York.

Dubinsky, Y., Kruchten, P., 2009. Software development governance (sdg): report on 2nd workshop. ACM SIGSOFT Software Engineering Notes 34 (5), 46–47.

http://www.eclipse.org/org/.

Farbey, B., Finkelstein, A., 1999. Exploiting software supply chain business architecture: a research agenda. In: Proceedings of the 1st Workshop on Economics-Driven Software Engineering Research (EDSER-1), Los Angeles, USA, 1999.

Gawer, A.G., Cusumano, M.A., 2002. Platform Leadership. Harvard Business School Press, Boston, MA, USA.

Gorschek, T., Fricker, S., Brinkkemper, S., Ebert, C., 2010. Third International Workshop on Software Product Management, Atlanta, USA. SIGSOFT Software Engineering Notes 35 (2), 25–29.

Hevner, A.R., March, S.T., Park, J., 2004. Design Science in Information Systems Research. MIS Quarterly 28, 75–99.

Iansiti, M., Levien, R., 2004. Strategy as ecology. Harvard Business Review 82 (3), 68–78.

Jansen, S., Brinkkemper, S., 2008. Applied multi-case research in a mixed-method research project: customer configuration updating improvement. In: Steel, A.C., Hakim, L.A. (Eds.), Information Systems Research Methods, Epistemology and Applications, 2008.

Jansen, S., Brinkkemper, S., Finkelstein, A., 2000. Business network management as a survival strategy: a tale of two software ecosystems. In: CEUR-WS Proceedings of the 1st International Workshop on Software Ecosystems, Washington, USA.

Jansen, S., Finkelstein, A., Brinkkemper, S., 2009. A sense of community: a research agenda for software ecosystems. In: Proceedings of the International Conference on Software Engineering, pp. 187–190.

Kabbedijk, J., Brinkkemper, S., Jansen, S., van der Veldt, B., 2009. Customer involvement in requirements management: lessons from mass market software development. In: Proceedings of the Requirements Engineering Conference, 2009 (RE '09), Atlanta, USA, pp. 281–286.

Ko, D.-G., Kirsch, L.J., King, W.R., 2005. Antecedents of knowledge transfer from consultants to clients in enterprise system implementations. MIS Quarterly 29, 59–85.

Le Berre, D., Rapicault, P., 2009. Dependency management for the eclipse ecosystem: eclipse p2, metadata and resolution. In: Proceedings of the 1st International Workshop on Open Component Ecosystems, IWOCE '09, ACM, New York, NY, USA, pp. 21–30.

Messerschmitt, D.G., Szyperski, C., 2003. Software Ecosystem: Understanding an Indispensable Technology and Industry. MIT Press, Cambridge, MA, USA.

Ministry of Information and Communication Technology, India, Draft Policy on Open Standards for Egovernance, 2008.

http://www.opendesign.org.

Popp, K., 2010. Goals of software vendors for partner ecosystems – a Pratitioner's view. In: Proceedings of the First International Conference on Software Business, Brussels, Belgium, pp. 185–192.

Rönkkö, M., Ylitalo, J., Peltonen, J., Koivisto, N., Mutanen, O.-P., Autere, J., Valtakoski, A., Pentikäinen, P., 2009. National Software Industry Survey. Helsinki University of Technology.

Rajala, R., Rossi, M., Tuunainen, V.K., 2003. A framework for analyzing software business models. In: Proceedings of the 11th European Conference on Information Systems, Naples, Italy.

Raymond, E.S., 1999. The Cathedral and the Bazaar, 1st ed. O'Reilly & Associates, Inc., Sebastopol, CA, USA.

Riehle, D., 2009. The commercial open source business model. In: Proceedings of the 15th Americas Conference on Information Systems, San Francisco, USA, Lecture Notes in Business Information Processing, pp. 18–30.

Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14 (2), 131–164.

Scacchi, W., 2007. Free/open source software development: recent research results and methods. In: Zelkowitz, M.V. (Ed.), Architectural Issues, vol. 69 of Advances in Computers. Elsevier, pp. 243–295.

Sherriff, M., Williams, L., 2006. Devcop: a software certificate management system for eclipse. In: Proceedings of the International Symposium on Software Reliability Engineering, IEEE Computer Society, Los Alamitos, CA, USA, pp. 375–384.

Simonin, B.L., 1999. Ambiguity and the process of knowledge transfer in strategic alliances. Strategic Management Journal 20 (7), 595–623.

Souer, J., van Mierloo, M., 2008. A component based architecture for web content management: runtime deployable webmanager component bundles. In: Proceedings of the International Conference on Web Engineering, Yorktown Hights, USA, pp. 366–369.

te Molder, J., van Lier, B., Jansen, S., 2011. Clopenness of systems: the interwoven nature of ecosystems. In: Proceedings of the third Workshop on Software Ecosystems, Brussels, Belgium, CEUR-WS, vol. 746, pp. 52–64.

van Angeren, J., Kabbedijk, J., Jansen, S., Popp, K., 2011. A survey of associate models used within large software ecosystems. In: Proceedings of the third Workshop on Software Ecosystems, Brussels, Belgium, CEUR-WS, pp. 27–39, volume 746.

van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., Bijlsma, L., 2006. Towards a Reference Framework for Software Product Management. IEEE Computer Society, Los Alamitos, CA, USA, pp. 319–322.

van der Schuur, H., Jansen, S., Brinkkemper, S., 2011. The power of propagation: on the role of software operation knowledge within software ecosystems. In: Proceedings of the Conference on Management of Emergent Digital Ecosystems (MEDES 2011), San Francisco, USA.

West, J., 2007. The economic realities of open standards: black, white and many shades of gray. In: Greenstein, S., Stango, V. (Eds.), Standards and Public Policy. Cambridge University Press, pp. 87–122.

Xu, L., Brinkkemper, S., 2007. Concepts for product software. European Journal of Information Systems 5, 531–541.

Yin, R.K., 2003. Case study research. In: Vol. 5 of Applied Social Research Methods, 3rd ed. SAGE.

Slinger Jansen is an assistant professor at the Department of Information and Computer Science at Utrecht University. His research focuses on software product management and software ecosystems, with a strong entrepreneurial component. Jansen received his PhD in computer science from Utrecht University, based on the work entitled "Customer Configuration Updating in a Software Supply Network".

Sjaak Brinkkemper is full professor of organisation and information at the Department of Information and Computing Sciences of the Utrecht University, the Netherlands. He leads a group of about twenty researchers specialized in product software development and entrepreneurship. The main research themes of the group are methodology of product software development, implementation and adoption, and business-economic aspects of the product software industry.

Jurriaan Souer is a PhD student at Utrecht University and business developer at GX Software, a company that specializes in web content management systems. Jurriaan's current research interests are development and implementation methods of web information systems.

Lutzen Luinenburg is a product manager at GX Software, a software company that produces web content management systems. Lutzen's research interests include web information system development methods and software product management.