Information and Software Technology 56 (2014) 1508–1519

Contents lists available at ScienceDirect

Information and Software Technology

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

Measuring the health of open source software ecosystems: Beyond the scope of project health

Slinger Jansen *

Department of Information and Computing Sciences, Utrecht University, The Netherlands

ARTICLE INFO

Article history:
Received 8 July 2013
Received in revised form 28 March 2014
Accepted 3 April 2014
Available online 19 April 2014

Keywords:
Software ecosystem health
Open source ecosystems
Software repository mining

ABSTRACT

Background: The livelihood of an open source ecosystem is important to different ecosystem participants: software developers, end-users, investors, and participants want to know whether their ecosystem is healthy and performing well. Currently, there exists no working operationalization available that can be used to determine the health of open source ecosystems. Health is typically looked at from a project scope, not from an ecosystem scope.
Objectives: With such an operationalization, stakeholders can make better decisions on whether to invest in an ecosystem: developers can select the healthiest ecosystem to join, keystone organizers can establish which governance techniques are effective, and end-users can select ecosystems that are robust, will live long, and prosper.
Method: Design research is used to create the health operationalization. The evaluation step is done using four ecosystem health projects from literature.
Results: The Open Source Ecosystem Health Operationalization is provided, which establishes the health of a complete software ecosystem, using the data from collections of open source projects that belong to the ecosystem.
Conclusion: The groundwork is done, by providing a summary of research challenges, for more research in ecosystem health. With the operationalization in hand, researchers no longer need to start from scratch when researching open source ecosystems' health.

1. Introduction

"Ruby or Python?" "SugarCRM or a closed-source competitor?" "Drupal or Joomla?" "RedHat or Ubuntu?" These are questions often asked by developers, professionals, entrepreneurs, architects, and stakeholders related to software producing organizations. Choosing between different ecosystems is a complex task and such a decision will be determining many of the future developments within an organization. At present the only way to answer such a question is by doing sufficient reading, asking around, and finding out what the risks are of choosing to enter an ecosystem. One indicator of whether an ecosystem is alive or not can be determined by looking at the health of the keystone project, for instance by looking at the activity surrounding the Ubuntu project. Such activity consists of commits, recent releases, fixes, number of downloads, response times in forums and bug trackers, activity on e-mail lists, and contributions from non-developers. However, project health ≠ ecosystem health.

Ecosystem health is operationalized in this work by taking a combined view at a keystone project and its surrounding projects. This work stands on the shoulders of two relevant contributions in the field of ecosystem health measurement. First, the work by Crowston et al. [3], who have provided a first operationalization of open source software project health, is used to establish health factors on the project level. Their work is also fundamental to OSSMole,[1] a collection of meta-data about projects in some of the main repositories, like Github and SourceForge. Secondly, the work of den Hartigh et al. [6], where an operationalization of health measurement of a commercial ecosystem is provided, is followed as closely as possible.

Software ecosystems are sets of actors functioning as a unit and interacting with a shared market for software and services, together with the relationships among them [15]. A healthy unit should thus express qualities typically associated with health:

* Tel.: +31 619884880.
E-mail address: slinger.jansen@uu.nl

[1] http://flossmole.org/.

http://dx.doi.org/10.1016/j.infsof.2014.04.006
0950-5849/

liveliness, activity, longevity, etc. For this work, we take a simple definition for software ecosystem health: *longevity and a propensity for growth* [19]. The definition is only the first step, as both longevity and propensity for growth can be operationalized in different ways with a plethora of different metrics.

There is a distinct need for an Open Source Ecosystem Health Operationalization (OSEHO). Manikas and Hansen [23] recently published a call to action for the creation of such an operationalization, and laid the groundwork for it. Also, in our research agenda for software ecosystems [17], we call for more research into ecosystem health. Others have attempted to create their own operationalization, but these typically get stuck in the concept phase [31,3,30]. In this article, an OSEHO is provided and evaluated using four research projects into open source ecosystem health.

We continue this work with a description of the literature on health measurement in ecosystems and open source projects. Section 3 discusses the creation of the OSEHO and its evaluation challenges. In Section 4, the OSEHO that provides methods for measuring health of open source software ecosystems is presented, consisting of a generic ecosystem health model and a set of methods for analyzing open source ecosystem health. In Section 5 four research projects are presented that apply parts of the model in practice. Furthermore, an analysis of the research projects and their aims (provide insight mostly), the indicators most frequently used (active developers, projects), and the research methods applied (mining repositories, web scraping) are presented. Section 6 presents a set of challenges that are met when applying the model and that were found in the four research projects, mostly having to do with data selection, preparation, and analysis. The article ends with a discussion on the applicability of an OSEHO and a summary of the conclusions and future research challenges.

2. Literature about ecosystem health

There is surprisingly little literature available about open source ecosystem health. Different perceptions exist and frequently ecosystem and project health are used interchangeably, such as in the work of Lundell et al. [20], who discuss open source ecosystems as being equal to one project. In the continued work of Gamalielsson et al. [9,8], the responsiveness of developers on the mailing list of the Nagios community is measured as an indicator for open source community health, but does not take an extended view of multiple projects within that community. The correct use of the term ecosystem in the open source domain is illustrated by Lungu et al. [21], who look at ecosystem as federations of systems. For this work the aggregated view is seen as the only way to look at ecosystem health, as the project level has already been studied extensively.

The research in this project is largely dependent on the proposal of Manikas and Hansen [23], who wish to operationalize the ecosystem health concept. Manikas and Hansen split their work along the categories of software ecosystems, business ecosystems, open source ecosystems, and natural ecosystems. In our literature survey the category of natural ecosystems is excluded, as this is already sufficiently discussed by Manikas and Hansen and also by Dhungana et al. [7], and a distinction is made between open source project health and open source ecosystem health. Related work is thus roughly divided into the categories: project health, commercial ecosystem health, and open source ecosystem health. The sources of literature are two literature surveys in the domain of software ecosystems [22,2] and in the domain of open source ecosystem health [23].

In regards to open source ecosystem health, the work of Wynn [31] first takes the three factors into account of vigor, resilience, and organization, analogous to natural ecosystems. The terms are later adopted and changed by Iansiti and Levien [14], to productivity, robustness, and niche creation. The framework presented by Wynn has been inspirational, and many of the factors in Wynn's framework have made it into the OSEHO presented in Section 4. Unfortunately, Wynn's framework does not present an evaluation or validation of the framework or its measures. The work of Manikas and Hansen [23] has also been fundamental, as it establishes an ecosystem as a collection of projects, and is taking different viewpoints into ecosystem health, thereby functioning as an evaluation checklist for the OSEHO. Finally, the work of Mens and Goeminne [25] (a detailed description of this work is given in Section 5), even though the word health is never mentioned in their chapter, is inspirational when looking at the collected metrics of developer roles and activity.

The work on project health has been used extensively: metrics about project health in most cases can be aggregated to the ecosystem level and thus most (if not all) project health metrics are relevant for the OSEHO. The works of Crowston et al. [3] and Wiggins et al. [30], operationalizing project health by looking at factors such as developer activity, have provided around a third of the metrics that can be found in the project level metrics of the OSEHO. A survey by Haenni et al. [11], in which developers are asked what they want to know about the software ecosystem in which they are active, concluded that developers have downstream needs: "what is the available public support? What licenses are used? What is the quality of other projects? What documentation is available?" and upstream needs: "What other projects use my project? How do these projects develop? How is my API used? Are code conventions followed?" These questions have been taken into account in creating the OSEHO framework as well.

Finally, commercial ecosystem health is a highly relevant topic. It is discussed extensively by Iansiti and Levien [14], who first provide guidelines on how ecosystem health may be operationalized. The work has been fundamental to the work of den Hartigh et al. [6], who first try to operationalize the health of a business ecosystem, based on the categories of health metrics presented by Iansiti and Levien. The challenges that Den Hartigh and his team face in the commercial domain (missing data, impossible to create one single health measure, etc.), have been essential to the design of the OSEHO.

3. Research approach

The goal of this research is to provide a comprehensive overview of the health metrics that can be used to determine the health of an open source ecosystem. It does so by creating an inventory of all metrics mentioned in literature that could potentially indicate ecosystem health and then placing these metrics in a framework. The framework can be applied by researchers who aim to reach a goal associated with ecosystem health, such as improve activity in an ecosystem, evaluate the health of one ecosystem over another, or identifying weaknesses in an ecosystem with the aim of making it healthier. The research answers the research question *"What are the health indicators for open source ecosystems and how can, if at all, the indicators be classified and operationalized?"* Please note the wider scope of the open source ecosystem: the research does not aim to evaluate the health of a single project, as that has extensively been done (see Section 2).

The evaluation of the framework is done by examining four research projects into ecosystem health, that have recently been done in the domain of software ecosystems. The four research projects have been selected from two literature surveys [2,22] and through searching for ecosystem health keywords. The selection criteria for inclusion are based on the fact that they take an ecosystem wide view, instead of just a project view, on open source

ecosystem health. The four projects have been described in four papers, and published in software ecosystem forums. No other fitting projects were found, and we consider it future work to progressively add new studies on software ecosystem health as they are published. The works that were selected were analyzed as follows: first, the aim of the research was extracted. Secondly, the sources that were used for data gathering were extracted and described, to present an overarching view of typical sources for ecosystem health studies, as described in Section 5.2. Thirdly, the methods for gathering data are inventoried, also as illustration to future researchers. Fourthly, and perhaps most importantly, the metrics that were collected are listed to establish their role and function in the OSEHO. No new metrics were added based on the papers, which furthers our belief that the previous works describing ecosystem health (without operationalizing) and works describing project health covered all the metrics already. Finally, the contribution, challenges, and discussions in the papers are analyzed to further illustrate the use of the metrics.

4. Open Source Ecosystem Health Operationalization (OSEHO)

Fig. 1 represents the OSEHO. The framework is built up out of three pillars, being the productivity, robustness, and niche creation pillars, which are addressed in the discussion of the literature in Section 2. The pillars are separated into three layers, being the theory level, the network level, and the project level. At the top level is displayed what the theoretical model of Den Hartigh prescribes to use as guidelines for operationalizing the health concept, which in turn is inspired by Iansiti and Levien [13], which in turn is inspired by the concepts from natural ecosystems: vigor, resilience, and organization. The translation into the open source domain is done on the second level, where the health operationalization at the network level is presented. At the third level a comprehensive overview is created of project health metrics, which, if collected for multiple projects in an ecosystem, can be used as an aggregate metric to describe overall ecosystem health.

4.1. Creating the OSEHO

The framework was created by first establishing that the split into the network level and project level is necessary to distinguish between ecosystem level metrics and (aggregated) project level metrics. Secondly, two lists of papers were created, one with project level health operationalizations and one with ecosystem health operationalizations. The metrics from each of the levels were collected. Then, to add more structure and make sure the essential elements of ecosystem health were covered, the three pillars of productivity, robustness, and niche creation were added, following the work of den Hartigh et al. [6].

Metrics were included when they fit the following criteria: (1) the metric had to stem from literature about project health or ecosystem health, (2) contribute positively to ecosystem health, (3) the metric was operationalized or at least operationalizable into a measurable entity, and (4) the metric was generalizable to multiple projects to get to the ecosystem scope. As the metrics came from literature, the second, third, and fourth criteria were added to filter out metrics that were not relevant, not contributive, or not operationalizable for the OSEHO. Please note that when a metric was reversible, only the positive variant of the metric was added, as for example "installs over a month" versus "de-installs over a month". The metric selection criteria are based on the source of metrics (criterion (1)), the definition of ecosystem health

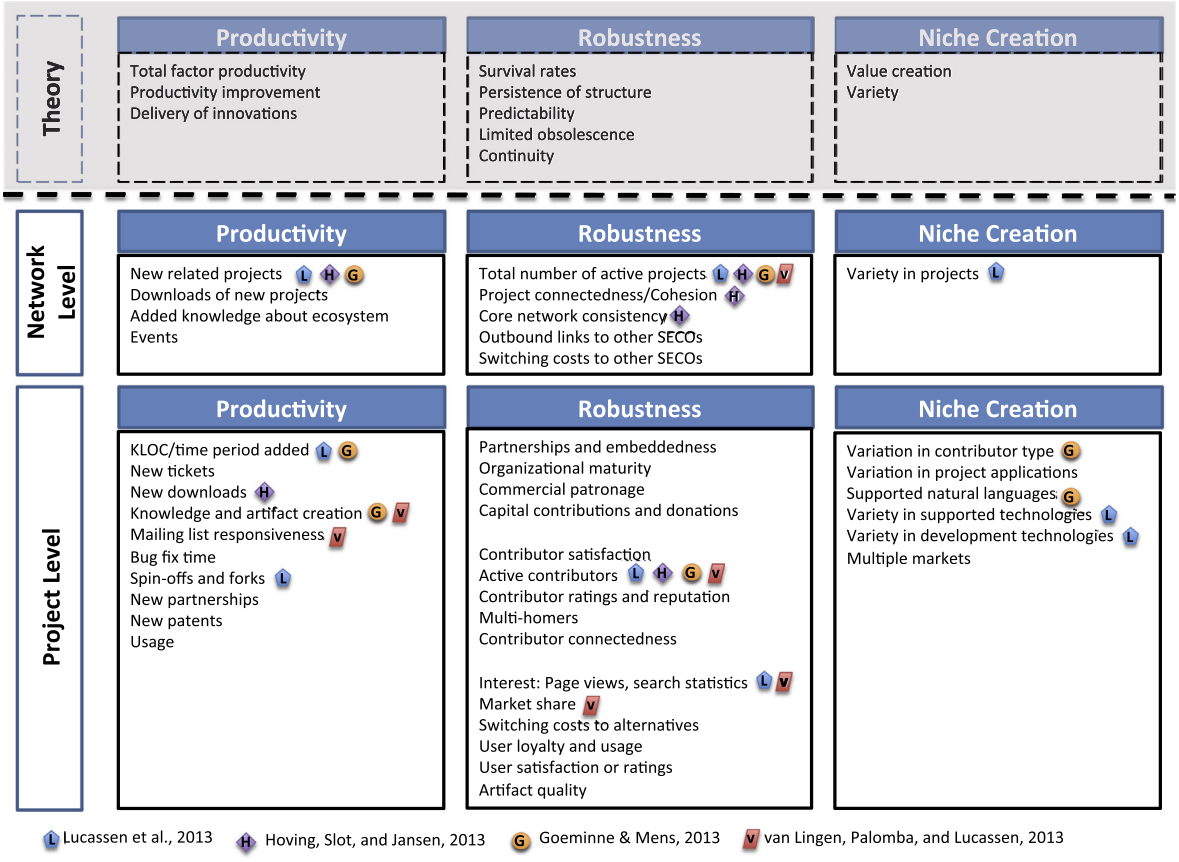

Fig. 1. Open Source Ecosystem Health Operationalization – the shapes indicate the occurrence of the metric in one of the research projects.

("longevity and a propensity for growth") (criterion (2)), and the requirement that the OSEHO has to be realistically applicable (criterion (3)) to multiple ecosystems (criterion (4))).

Metrics that particularly address growth, such as *new related projects*, are put into the productivity column. These metrics should specifically address a variable change over a time unit (new projects per month, new commits per week, etc.). Metrics that address robustness, such as *the number of active projects*, are added to the robustness column. These metrics are typically used to compare ecosystems in terms of size ("as of November 2013, the Python ecosystem contains at least 39,000 extensions, whereas the Ruby ecosystem contains at least 66,000 extensions, so arguably the Ruby ecosystems is larger"). Some metrics are about variety within the ecosystem, such as *contributor types*, *project variety*, and these are put into the niche creation column. We did not encounter metrics that fit multiple pillars in the system. The metric "cohesion", for instance, is a static measure and therefore fits the robustness column. The metric "new partnerships" addresses a change over time, and therefore fits the productivity column.

There is little overlap between the columns, as the robustness column addresses absolute entities, whereas productivity metrics typically address changes over a period of time. The niche creation column also addresses absolute numbers but has little overlap with the robustness column, as it addresses specific variables that mostly deal with variability. There are many potential dependencies between metrics, however, there exist little "hard" dependencies. For example, a project that has many *happy contributors* may have more *active contributors*, but these relationships are situational, as a mature ecosystem may not require large numbers of contributions from its developers. In a similar example, there might be many more *new projects* in an ecosystem with many *different markets*, but if the ecosystem is not healthy, there may be no new projects, despite being active in different markets.

4.2. Network level

A plethora of work has been done on software project health. However, as concluded previously, it is impossible to directly project the health of a project onto ecosystem health, as ecosystem health takes into account multiple interrelated projects, contributors, and end-users. The network level of the OSEHO concerns metrics that describe ecosystem health and can only be described on that level. The *events* metric, for instance, looks at organized events where stakeholders are brought together that share an interest in the total ecosystem, instead of only including *specific project events*.

The **Productivity metrics on the network level** are those metrics that indicate the productivity in the ecosystem. First, *new related projects* are projects that are part of the ecosystem, such as the launch of a new plug-in for the Eclipse development environment. The *downloads for new projects* indicate that the ecosystem is healthy, as new projects still find a welcoming community. The *added knowledge* indicates that contributors are adding knowledge, such as aggregated information, blog posts, and manuals, indicating also that the ecosystem is healthy. Finally, organized *events* surrounding the ecosystem are an indicator that the ecosystem is healthy. Events are a measure of productivity, as an increase in events over a certain time period may indicate a more healthy ecosystem.

The **Robustness metrics on the network level** indicate how well the ecosystem will deal with change and how quickly it will recover. The *number of active projects*, for instance, is a strong indicator of strength and power in numbers. *Cohesion*, i.e., connectedness between members, and *core network consistency* are both indicators of how well connected parts of the network are, based on the assumption that a well-connected network is healthier. *Outbound links to other software ecosystems* indicate how well the ecosystem is connected to other ecosystems, and how much those ecosystems depend on this ecosystem. Outbound links are a measure of robustness, as a more connected ecosystem can better withstand shocks within the ecosystem by enabling well connected projects to seek activity, revenues, and end-users from other domains, ecosystems, and projects during such a shock. *Switching costs to other ecosystems* indicate how hard it is to move to another ecosystem for an ecosystem player, and how easy it is to multi-home as a niche player, i.e., avoid strong dependencies on an ecosystem by being part of multiple.

The **Niche Creation metrics on the network level** describe how much opportunity there is in the ecosystem to start as a new niche player. This is mostly defined by *a large variety in projects*, indicating there are many niches, platforms, domains, etc., in which a new player can become active.

4.3. Project level

The network level describes those metrics that can only be calculated by looking at the complete ecosystem. One level deeper, one can look at the ecosystem's constituents (i.e., the projects and everything attached to those projects). The project level metrics are to be seen as metrics about the aggregate metrics that can be derived by looking at multiple projects. Single project metrics are not interesting, unless the project constitutes a significant portion of the ecosystem.

The **Productivity metrics on the project level** are the metrics that indicate how much the projects contribute to the total ecosystem. The most important metric is typically *the number of lines* of code added or changed over a time period, as this is a prime indicator for the activity of contributors in the ecosystem. Productivity is also seen in the *knowledge and artifact creation* by contributors, such as manuals, translations, marketing materials, etc. Also, the number of *spin-offs and forks* provides a relevant metric, as this indicates developer interest. Finally, *new partnerships* and any *patent activity* indicate that the project, and thus its surrounding ecosystem, is becoming more productive.

Other indicators are *the number of tickets* and related *bug fix time*, as these indicate how quickly problems are reported and resolved in the ecosystem. Similar to this is the *livelihood of a mailing list*: the more lively it is, the more people are (probably) contributing and using the project. The *number of downloads* indicates how many end-users actually start using the product, and any increase or decrease in this number is an indicator that health may be increasing or declining. The *usage* of a project is also highly indicative of how important a project is to its end-users, and can be used as a predictor of robustness.

The **Robustness metrics on the project level** indicate how well the project deals with change. Typically, the best defense for a project to survive a big change is in numbers: a large number of *active developers*, for instance, is indicative of a robust project, and the more robust projects there are in an ecosystem, the healthier it can be considered.

Because of the large number of metrics in this category, the indicators are put into three conceptual groups: organizational, contributors, and end-users. The three groups are mostly created for clarity. They do, however, enable a researcher wishing to research the robustness on a project level, to address one of the groups separately with a particular research approach. The contributor group, for instance, can be approached by a survey to obtain the metrics that are relevant in that group.

The organizational metrics include *partnerships and embeddedness*, as partnerships indicate a strong project that is well embedded in the community. Similarly, *commercial patronage* and *capital contributions and donations* are indicators of acceptance by commercial organizations, also showing that the project will probably not go away soon. Finally, *organizational maturity* of the pro-

ject indicates that the project is probably managed well, thereby also guaranteeing robustness for the project.

In the group of contributors, the most important metric is the number of *active contributors* and their *satisfaction* with the project. Another factor that can be taken into account when establishing how robust an ecosystem is, are *contributor ratings and reputation*, describing how well the developer is contributing and performing within the ecosystem. These ratings and reputation can be established by looking at their standing in the community, their numbers of commits, their individual bug fix times, etc. When these developers *multi-home*, it adds to the robustness of the project as the developer is well connected and may establish new relationships for the project. A final sign of robustness for developers is that a contributor is *well-connected* in the project, as highly connected networks are more robust than loosely connected networks.

Finally, in the end-user group, *end-user interest*, in the form of page views, project "followers", search statistics, and other indicators of interest can be used to establish robustness. Also, *market share* and *user satisfaction* are clear indicators that tell whether a project is robust or not. Furthermore, if *switching costs* to other solutions are high, end-users tend to stay with a project for a long time. Also, *user loyalty and usage* indicate how robust a project may stay in the future, if metrics such as usage and attrition are known. Finally, the *quality of the project artifacts* indicate whether the project is fickle, or whether it has been built to last.

The **Niche Creation metrics on the project level** indicate whether the project allows for sufficient freedom and variation to enable end-users and contributors to create new niche solutions with the project. When a project, for instance, *supports many different technologies* it can be used broadly in different contexts, thereby facilitating new and innovative ways of using the project. The same holds for a project that is used in *different markets*, is *available in different languages*, and has been built with *different development technologies*, as that allows more people with different skill sets to join in with the project. From these factors it already becomes clear that having a project that can be applied in a *wide variety of contexts*, will be more supporting for niche creation. Finally, a strong *variation in contributor types* shows that a project is mature and enables different types of contributors to come up with new domain specific applications.

Manikas and Hansen [23] in their work look at actor health, software health, and orchestration health. In the OSEHO, these levels are not identified explicitly. The actor in "actor health", for instance, may be the open source developer, the organization that developer stems from, or even the keystone organization itself. In the OSEHO most of the actor health factors are part of the project level. Orchestration has deliberately been left out of the OSEHO: the results of orchestration are reflected by changes in the metrics in the OSEHO and should therefore not be part of the OSEHO, in our opinion. One aspect of orchestration, for instance, is licensing. A license can be detrimental to the adoption of a project, if it is infectious [28], for instance. Should a license be changed to become more reuse friendly, this should be reflected in the metrics over time. Finally, the OSEHO spends less attention on software health, whereas Manikas et al. specifically address software component health, platform health, and software network health. In the OSEHO these are not specifically addressed, but are reflected in the artifact quality measures, project connectedness metrics and in the variety measures.

4.4. Analysis method

When a researcher is interested in establishing the health of an open source ecosystem, she can apply the framework in the following way:

1. **Set goals –** First, the researcher will establish the goal of the health assessment. By summing up the goals, she can determine which metrics are most relevant. This is also the phase where the researcher will determine the frequency of data collection, so whether this is a one-time assessment or something that needs to be done continuously.
2. **Select ecosystem scope –** Secondly, the researcher needs to establish whether to study one subsystem in its context, or multiple ecosystems in a broader context. Scoping too narrowly results in a limited view on the complete ecosystem (ignoring the non-open source browser Internet Explorer when deciding for which browsers a browser plug-in will work, for instance), where a scope that is too wide results in non-informative data on larger ecosystems (gathering data on all Linux variants, where the subcluster of Debian variants was already selected as being most interesting for a specific research question).
3. **Select metrics –** The goals are used to select the specific metrics that are relevant to reaching those goals. Preferably, the metrics can be collected comprehensively, but as will be highlighted in the research challenges in Section 6, this is not trivial and practically impossible in most cases, so the researcher will have to suffice by collecting data for subsets of the metrics.
4. **Assess available data –** The researcher will assess which data is available and whether the collection will be sufficient to reach the goals set in the first step.
5. **Collect data –** The data is collected using the most efficient methods available, such as repository mining, sending out a survey to ecosystem participants, doing financial analysis, studying literature and existing reports, and any other method that can satisfy the data requirements described in step 2.
6. **Analyze the data –** The data is analyzed and goals are satisfied.

The analysis does not differ from a typical data mining project and must be customized for each ecosystem or research project. A relevant question that must also be asked is how frequently one wishes to perform the analysis. Furthermore, it is recommended that any tools or data sets are published, as they provide a history to future researchers, can be used to validate research, and can be reused in case the project is redone in the future.

In scoping the ecosystem, a fitting view on software ecosystems is taking the view that the complete software ecosystem is a large database with organizations, ranging from one-man teams up to multi-national software vendors, that produce software. When deciding which of these organizations should be included in the ecosystem analysis, the researcher should define inclusion criteria, such as "creates Linux software", "creates plug-ins for Firefox", or (even narrower) "writes scripts for GreaseMonkey (a language for changing how web pages are viewed in the Firefox browser)". As soon as the set of organizations that are included in a research project is known, the researcher can start collecting data on them.

4.5. Applying the method and OSEHO in practice

When applying the method, beneficiaries need to establish their goals. To illustrate, we provide the following non-exhaustive list of use cases for the method and the OSEHO.

In the first and perhaps most important scenario, an open source project organization must constantly look at its own growth and health. Several examples of metrics that must be continuously collected and evaluated are the number of third-party extensions to the platform, the number of active contributors, the events organized around the world for the project, download numbers, and success metrics of other competing projects. It is interesting to

see that an organization such as the Eclipse foundation consists mostly of ecosystem coordinators, intellectual property experts, and support engineers.[2] Furthermore, it should come as no surprise that the Eclipse Foundation in their year end reports focus specifically on the metrics mentioned above.[3]

In a second scenario, software producing organizations regularly need to assess whether they will have their commercial activities depend on an open source platform, such as a database platform, a web server, an IDE, or another essential third-party component. These decisions are typically long-term: the software producer expects to benefit from a strategic decision about depending on a third-party platform and must invest to integrate that component. Such a decision should not be made lightly: what if the third party platform changes its license? Or what if the team behind the component decides to fork a project and continue under a different name? And what if important issues are rarely fixed? Software producing organizations must regularly (i.e., yearly) do a health check of the ecosystem surrounding the platforms they depend on, as radical changes in these ecosystems can strongly influence the success of the software producer's software products. In several case studies we have observed that the software producing organization started participating actively in the development of the open source platform or started themselves building plug-ins for it [16].

A third scenario is that of an end-user representative deciding whether to structurally start using an open source platform, such as for Word processing, for long term use. An end-user representative can potentially decide for thousands of users whether to go for an open source platform (i.e., LibreOffice) over a commercial variant. Such a choice is obviously not made lightly: a platform needs to be adopted into an organization's infrastructure, providing mass deployment, support, courses, and other services surrounding the platform. The application owner within that organization must frequently check on the health of the ecosystem surrounding the project, to find potential extensions for the platform, to see how competing platforms are doing, and to make sure there is a fertile and healthy ecosystem surrounding the project to ensure of its continuance. The health scan will in this case typically serve as an addendum to a much heavier functional evaluation of the platform.

Putting to use the OSEHO, organizations can gain strategic advantage over others. Software ecosystem coordinators (such as the Eclipse Foundation) require the information that stems from the metrics in the OSEHO to make well-informed decisions about their strategy. If we (imperfectly) equate a software ecosystem to a platform, we can follow Cusumano's Staying Power [5] conditions for a successful strategy. Specifically, he mentions that there should be strong network effects, there should be little differentiation between platforms, and multi-homing must be hard. Many of the metrics can assist in making decisions about these three factors. The more active contributors there are, for instance, the more network effects you gain. Furthermore, the more domain specific solutions there are to your platform, the more end-users can use the platform. In regards to multi-homing, it is interesting to see for what other platforms the extenders in your own ecosystem release their component: it may be beneficial to make multi-homing harder or to assimilate the other platform into the ecosystem. With the metrics in mind, ecosystem coordinators can make strategic decisions about where they want to improve and invest.

The OSEHO attempts to do the same for end-users and end-user organizations: provide them with insight into the healthiest ecosystems such that they can adjust their strategy to it. The OSEHO enables an end-user organization to choose, for instance, between investing in the development of a domain specific solution for a healthy platform, versus taking what is already available from an unhealthy platform. In a long term strategy, the first option should prevail.

5. Analysis of the research projects

Four research projects have been selected to illustrate the use of the OSEHO. The selection criteria have been listed in Section 3.

The first project applies ecosystem health metrics to determine how healthy the ecosystems surrounding commercial Platform as a Service providers are [19]. The goal was to provide stakeholders in these ecosystem with insight into their ecosystem development and the most important metrics that indicate success in these ecosystems. The data source was GitHub and statistical analysis was used to determine that Heroku, a platform as a service provider, is currently doing best, but that there are others growing very quickly.

The second project looks into the health of the Python open source ecosystem, by extracting information from a collection of projects that are part of the Python ecosystem [12]. The Python ecosystem is highly active and grows quickly (exponentially, at some points even). The stakeholders in mind are keystones in the ecosystem, as they may wish to steer the ecosystem in a certain direction. A number of key characteristics defining the health of an open source ecosystem are established and the authors call for an extension of the work to other ecosystems.

The third project aims to validate a framework for open source community analysis. The framework is still in use and further developed, with the aim of analyzing developer behavior in open source communities, where health does not have to be the main focal point in the research. They evaluate the framework by taking an extended view at all projects in the Gnome ecosystem [25]. They establish the roles and activities of developers over the different projects in the ecosystem.

The fourth project looks at three of the largest open source content management systems and finds that ecosystem health and adoption are not equal [27]. Health is looked at by extracting project information from projects related to the content management systems. The largest challenge found in the paper lies in the health comparison between the three ecosystems. The research projects are summarized in Table 1. For each of the rows in the table, the observations about the projects are discussed.

5.1. Aims, research methods, frameworks, indicators, and contributions

When looking at the **aims** of the different research projects, it becomes clear that the recurring pattern is that some other entity, outside of the academic community, can be informed by the knowledge that is collected about the ecosystems under study. The four projects mention that two groups are going to profit from the results: (1) keystone players that need information about their ecosystems and (2) stakeholders that are (planning to be) part of an ecosystem. Two of the projects focus on one ecosystem and thus get in-depth knowledge about the platforms, whereas the other two do a comparative study of different ecosystems to compare and contrast. The OSEHO has a third target group: to provide researchers with a framework for establishing ecosystem health and potentially even to extend the framework with new metrics.

The **research methods** applied show an interesting mix of qualitative and quantitative data gathering and analysis, with a focus on the latter. Furthermore, the four projects display the use of multiple data analysis methods, indicating that multiple sources of

[2] http://www.eclipse.org/org/foundation/staff.php.
[3] http://www.eclipse.org/org/foundation/reports/annual_report.php.

Table 1
Research project survey.

Title	P1: Ecosystem health of cloud PaaS providers	P2: A study of the health of the Python ecosystem	P3: The Case of the Gnome Community	P4: Comparing the ecosystem health of CMSs
Publication	Lucassen et al. [19]	Hoving et al. [12]	Mens and Goeminne [25]	van Lingen et al. [27]
Aim	Evaluate the health of Platform as a Service providers in a cost effective and objective manner	Evaluate the health of an open source ecosystem and identify typical developer collaboration types	Validate a framework for open source community (health) analysis and provide insight into the Gnome community	Establish health of three ecosystems to inform end-users or niche players of the most successful CMS
Sources	GitHub. BitBucket, Tigris, and LaunchPad were discarded due to data scarcity	The Python index of components: http://pypi.python.org	The Gnome project list: git.gnome.org/browse/	The component overviews of three ecosystems, Wordpress.org, Joomla.org, and Drupal.org. Forum posts. Google for google hits per project, and Google Trends
Method	Select platforms and data sources, mining app using GitHub API, data preparation, 1% manual cross validation, some false positives found	Download all developers from manifests, split up developer lists, identify relationships *created_by* and *collaborated_with*	Mining detailed commit data in the Gnome Community, by analyzing all projects on the Gnome project sites	Data gathering through a survey, an automated scraping tool, and some manual work.
Health frameworks	(A part of) Crowston	Parts of den Hartigh et al., Wynn et al. and Iansiti et al.	Health is not specifically mentioned in the article	None
Indicators	**Active contributors:** active developers in the past year, active developers of unique repositories in the past year, active developers per segment of time, lines of code added per period of time. **Spin-offs:** total repositories, unique repositories, forks. **Interest:** number of followers, unique programming languages, multi-platform repositories, repositories updated at least once, active repositories	Active contributors, number of projects, number of total project downloads, growth in number of projects, project connectedness	Active contributors, number of authors. KLOC/time period, contributor activity, number of projects, project connectedness, variety in roles, sub-communities, activity division across sub-communities, do community members specialize over time	New related projects, active contributors, Up-to-dateness of projects, "findability" of the ecosystem, centrality of the platform, market share analysis, level of contribution per community user, perceived ecosystem health
Contribution	The research provides a method for quickly providing insight into both internal and external stakeholders in a PaaS ecosystem and its health. Furthermore, it shows how ecosystem health can be used outside the traditional ecosystem views	Provides a snapshot of the Python ecosystem, which successfully exhibits an increasing growth in numbers of components added to the ecosystem. It also discusses how connectedness could be improved to make the ecosystem healthier	Exemplary analysis of sub-communities, community members specialize over time, overall activity appears to be dropping, coding remains the activity that requires most effort, although other activities become more important over a project's lifetime	A status report on the health of three ecosystems, ecosystem health $\neq$ platform success
Challenges	Data filtering, data unification, download count not working on GitHub. Some components on GitHub support multiple platforms. Operationalization of indicators of Crowston et al. [3], specifically active developers, spin-offs, interest	Expand the work by taking a closer look at the roles that developers take on (bridgers, lone wolfs, brokers, etc.). Continuous measurement. Detailed project data, such as code, must be retrieved from different sources (SourceForge, GitHub, etc.) Missing data, even from the manifest files	Data unification, advanced architecture developed to approach the problems posed by distributed data, heterogeneous data, etc.	Historical data gathering impossible, plethora of research methods necessary to gain a balanced view
Discussion	Absolute numbers should not be leading, developments in some smaller ecosystems are healthy and strong. No financial data available for each platform. Continuous data gathering	More in-depth study needed before a full report can be given	None	Hard to compare three somewhat equal ecosystems, Google search statistics not always reliable, no historical data

data are typically used in ecosystem health research. The work of Mens and Goeminne [25], calling for a generic data collection and analysis workbench for ecosystem analysis, has an obvious role here, as the other three projects are using interesting mixes of web scraping, failing APIs, and manual data gathering. In the project studying content management systems (CMSs), the data is validated in part by conducting a survey among niche players in the CMS ecosystems.

The **frameworks** that are used for outlining the health research are mostly developed by the authors themselves. Lucassen et al. [19] use a part of Crowston's framework, and Lucassen et al. [12] refer to the works of Iansiti and Levien, den Hartigh et al. and Wynn. These frameworks can be considered comprehensive, but not necessarily for the domain of open source ecosystem health. Furthermore, even though the work of Wynn for instance is comprehensive, it is not operationalized, further strengthening the need for the OSEHO. The OSEHO strongly bases itself on these frameworks, but can be considered more extensive, because other works than the ones used in the research projects have been included in the evaluation as well, such as the work of Wiggins et al. [30] and that of Haenni et al. [11].

The **indicators** mostly used are contributors, projects, activity, relationships between contributors, and interest and are found in the four research projects. The contributors, projects, and interest are typically easy to obtain. Contributors are typically gathered through the manifests of projects, which in turn are gathered from project lists. Interest is gathered using Google Trends for two of the research projects and by looking at secondary variables (forks, followers, etc.) in the other two research projects. Relationships are harder to obtain but are typically determined by looking at co-authorship or collaboration on similar projects and code. Furthermore, it appears that the research projects were strongly influenced by the availability of data: the research design was typically created with the available data in mind. Please note that some of the indicators in the projects are operationalized versions of the metrics in the OSEHO. Take for instance the indicators "up-to-dateness of projects", "findability of the platform", and "centrality of the platform" from the project of van Lingen et al. [27]. "up-to-dateness of projects" is an operationalization of "total number of active projects", "findability" is an operationalized version of "search statistics", and "centrality of the platform" is an operationalization of "core network consistency". In the OSEHO a degree of freedom is necessary, as these indicators are just one way of interpreting the metrics. Other interpretations are possible, hence there is not an exact match between the indicators found in the four research projects and the metrics in the OSEHO. The OSEHO does, however, encompass the metrics used in the research projects.

The **contribution** of the work by Lucassen et al. [19] provides a report on the Cloud providers and the surrounding activities on GitHub. Furthermore, the work shows how GitHub can be used to gauge health of commercial or closed ecosystems. The work of Hoving et al. [12] provides insight into the Python ecosystem and attempts to provide stakeholders with tools to increase connectivity. The work of Mens and Goeminne [25] gives an insightful view of the Gnome ecosystem, with a specific focus on contributor roles (such as translation, design, and coding activities) and attempts to provide a standard workbench for open source ecosystem analysis. The project of van Lingen et al. [27] compares the health of the ecosystems of three large open source CMSs and illustrates that ecosystem health (Drupal is the healthiest) does not equate platform success (Wordpress is more successful by far). The works succeed in providing ecosystem stakeholders with new insights into their communities. The works presenting new methods for data analysis are most interesting for the academic community, such as using Github for analyzing somewhat closed ecosystems [19] or introducing a framework for open source analysis [25].

What these projects show is a plethora of approaches in obtaining data on ecosystem health and subsequently analyzing it. The OSEHO aims to bring these approaches together and put them into context. These approaches furthermore illustrate the use and applicability of the evaluated aspects in the OSEHO and indicate that the OSEHO is applicable and useful for those aspects.

5.2. Data sources for open source ecosystem health

There are roughly three types of data sources for performing ecosystem health research. First, there are the *project sites*. Project sites are sites where all data about an open source project are collected, typically including the source code. Project sites are typically hosted by project hosting services, such as SourceForge, Github, BitBucket, or Tigris. Many of these project hubs are well aware of the wealth of data that is held in their databases. GitHub, for instance, has an advanced API that can be queried by anyone and SourceForge provides their data as one downloadable database, that can be reused for research purposes.

The second type of data source are *ecosystem hubs*. Such hubs contain essential project indexes, such as RubyForge's gem index, the Python egg index, the Joomla Component index, and many others. These sources are typically managed by a keystone organization within the ecosystem and are an excellent starting point for any ecosystem health project. Please note that the ecosystem project lists can also be collected by querying the project sites, but this introduces new challenges in regards to the elimination of false positives (see Section 6).

The third type of source are *aggregation sites*, where aggregated information is stored about an ecosystem, or even about all ecosystems. Ohloh.net stands out as one of the sites where information about as many open source projects as possible is collected. Another interesting source in this regard is StackOverflow, a questions-and-answers community for developers. One example of an application of data analysis of StackOverflow is the language popularity index, developed at Delft University.[4]

Besides these sources that can be scraped, downloaded, analyzed, and called upon with an API, it is always possible to perform a developer or contributor survey. This has proven successful in several ecosystem research projects, such as the CMS project [27] or a project into clusters in the Ruby ecosystem [26], where contributors were asked whether they are aware of their place in a cluster of contributors or not, and how these clusters were shaped or formed.

Three of the projects focus on project indexes created by the keystone players. The keystone obviously can play a crucial role here: without such indexes the community does not have a central gathering place for ecosystem participants. The fourth project has focused on Github as the central repository, after discarding several others, and used its search results as the way to getting related projects.

Data sources of high quality are essential for ecosystem health studies, on three levels. On the first level, there is a need for access to the source code of projects, in a generic way. The API of GitHub.com, for instance, is an excellent point of access for ecosystem researchers wishing to delve deeper into source code, for instance to study API adoption or code quality. On the second level, project indexes are required that summarize which projects belong to a certain ecosystem. These lists do not necessarily need to be complete, but provide starting points for researchers in ecosystem health. On the third level, data aggregation sites, such as Ohloh.net, provide researchers with high-quality secondary data, enabling for instance the study of developer migratory patterns and developer productivity across ecosystems. In the future it is expected for these secondary databases to flourish, such as for instance the

[4] http://langpop.corger.nl/.

Table 2
Data sources for collecting metrics.

Metric/source	Project sites	Ecosystem hub	Aggregation sites
Productivity			
New related projects		Project indexes	Project indexes
Downloads of new projects	Download page		
Added knowledge about the ecosystem		Content management system, books, wikis	
Events		Content management system	
KLOC added	Software repository		Contributions
New tickets	Ticketing system		
New downloads	Download page		
Knowledge and artefact creation	Software repository, content management system		
Mailing list responsiveness	Mailing list		
Bug fix time	Bug trackers		Bug trackers
Spin-offs and forks	Project repository	Projects history	Repository history
New partnerships	Information pages, code dependencies	Partnership model	
New patents		Developer survey	
Usage	Software operation knowledge		
Robustness			
Number of active projects		Project indexes	
Project connectedness/ Cohesion	Information pages, code dependencies	Partnership model	
Core network consistency		Partnership model	
Outbound links to other SECOs	Code dependencies	Partnership model	
Switching costs to other SECOs		Developer survey	
Partnerships and embeddedness		Partnership model	
Organizational maturity		Content management system, partnership model, rules and regulations	Explicitness ecosystem
Commercial patronage	Partnerships, content management system	Partnership model	
Capital contributions and donations	Partnerships, content management system	Partnership model, content management	
Contributor satisfaction	Developer survey	Developer survey	
Active contributors	Repository	Repositories, developer survey	Repositories
Contributor ratings and reputation	Project contribution size	Contributor aggregation	Rating systems, contributor aggregation
Multi-homers			Contributor aggregation
Contributor connectedness			Contributor aggregation
Interest	Search engines, page hits	Search engines, page hits	
Market share	Content management system, end-user surveys	Content management system, end-user surveys	Code inclusion in other projects and ecosystems
Switching costs	End-user and developer surveys	End-user and developer surveys	
User loyalty and usage	End-user and developer surveys, software operation knowledge	End-user and developer surveys	
User satisfaction or ratings	End-user surveys	End-user surveys	
Artifact quality	Code quality in repositories		
Niche creation			
Variety in projects		Project indexes, content management system, multi-homing	
Variation in contributor type	Contributor data, contributor surveys	Contributor data, contributor surveys	Contributor data
Variation in project applications	Project dependencies, content management system	Project dependencies, content management system	
Supported natural languages	Content management system, repository		
Variety in supported technologies	Project dependencies, content management system	Project dependencies, content management system	
Variety in development technologies	Project dependencies, content management system, repository		
Multiple markets	Project dependencies, content management system	Project dependencies, content management system, end-user surveys	

GHTorrent project [10], which is attempting to collect and maintain as much data from Github as possible, without having to stress the Github API and without running the risk of losing historical information. Recently, the GHTorrent project has become redundant, as Github is now making its data available through the Google BigQuery initiative on the GitHub archives site.[5]

In Table 2 we have inventoried the data sources for each of the metrics, categorized by the different levels on which data is gathered in ecosystems. The table shows that a large variation of data sources is required and available for measuring open source ecosystem health. The main sources are the repository, developer aggregate data (such as the developer's other projects, characteristics, and contribution size), the project indexes, the content management systems of the ecosystem hub and its projects, and any supporting systems for a project (bug trackers, mailing lists, etc.).

[5] http://www.githubarchive.org/.

One example is *market share*. To get *market share* of a project, we need to do end-user surveys and collect the knowledge that is already available. On the ecosystem level, we can also do end-user surveys and collect information that is already available, such as market reports, open source evaluations, and other platform popularity data. Finally, on an aggregate level we can analyze, using source code and manifest analysis, how frequently a project is required and used by other projects.

6. Repository mining research challenges

The research challenges from the projects are listed in Table 1 and are collected and summarized to form common research challenges into a research agenda. Each of the terms in bold can be considered a challenge for any new ecosystem (health) study that involves repository mining. The challenges are split into data selection challenges and data preparation and analysis challenges.

6.1. Data selection challenges

When starting a research project for the analysis of open source software ecosystems, the first step involves, after formulating a research goal, **data selection**. Based on the available resources, a project starts by selecting the data that is relevant to reach the research goal. The research projects report the following challenges in this research phase.

All projects report on the **absence of data**, such as missing project manifests or missing lists of authors. Such missing data forces researchers to remove data items from data sets, thereby reducing reliability of the final conclusions. The problem of missing data is especially painful when researchers wish to compare projects (apples and oranges). It can be extremely hard to compare, for instance, the health of two software projects based on mailing list response time, when one of the projects does not have a mailing list. Another example is given by forks in projects, i.e., when a developer decides to copy the source code branch and continue a new version. These forks are uncommon in subversion projects, not made explicit and typically hard to merge. For a versioning system like Git, however, forks are one of the most common ways to develop new features, so comparing the number of forks of a subversion project and of a Git project is pointless.

Besides it being hard to compare projects with different data, it is even harder when **comparing ecosystems**. If one of the ecosystem keystones stores different meta-data about projects than a stakeholder from another ecosystem, comparison becomes nigh impossible. Looking at Python versus Ruby, for instance, it is observed that Python reports on the number of downloads per component (egg), whereas Ruby does not maintain such a metric reliably, as many of its components (gems) are hosted on Github, where the download count metric is unreliable.

Another problem in the problem realm of missing data, is **the lack of historical data**. Establishing the health of an open source ecosystem, for instance, becomes much more interesting when looking at a developmental picture, using timelines, historical download data, commit data, etc. This data is, however, rarely available as project sites tend to store only current data. There are countermovements against this loss of information, such as (again) the GHTorrent project [10], which stores historical data about the event logs from GitHub, knowledge that would otherwise be lost.

Another problem for data selection is **project findability** and **ecosystem transparency**. When an ecosystem does not maintain a central index of projects, it is almost impossible to say anything sensible about the number of projects that are related to the ecosystem. A similar problem is that many of these projects are managed opaquely, i.e., by one organization or developer that does not share its source code. This is perhaps the biggest weakness of the Platform Comparison project (project 1), as some of the platforms may typically be used by more closed organizations (i.e., partners of Microsoft that develop for the Azure cloud), thereby making a large part of the ecosystem hidden.

A search challenge frequently mentioned is the **elimination of false positives**: as lists of open source projects are collected, for instance from Github, these projects are typically obtained by running a search query on the GitHub site. Several of the hits may concern projects outside of the desired scope, but these still mention the search terms. Or, in the case of Lucassen et al. [19], some open source projects were mentioned that can be used for several of the platforms and thus are part of the analyses for both the platforms.

6.2. Data preparation and analysis challenges

The four projects report on **incorrect data**, with perhaps the most emphasis on the missing of the number of downloads from GitHub, which is available through the API, but is a field that is incorrectly filled in. As Github and its API are relatively young this is not surprising, but has hampered several of the projects. In the future, an increase in data quality would strengthen ecosystem health research and we hope that open source project authors take the time to validate the data that is published about projects online.

A similar challenge is that of **data clean-up and preparation**. This may, for instance, involve splitting up contributor names and identities from the author field in a manifest (e.g., "Google Python Team and Guido"). Although the technical challenge of splitting up such fields by keywords 'and' and ',' is trivial, the next step of identifying and gathering of identities is much more challenging. Another problem that arises here is the challenge of **data merging**. It is impossible without extra information to for instance determine that 'DHH@37signals.com' is the same as 'david@37signals.com'. The good news is that initiatives such as Ohloh.net are quickly becoming central tomes of knowledge where data is united on open source projects, contributors, and their supporting organizations.

In the domain of data analysis, a common research challenge is found, being the **cross validation of results from different research methods**. In the project on CMSs, for instance, a survey was held amongst contributors about the popularity of the CMS. The survey data was then used in the analysis to strengthen the quantitative data, creating alignment challenges between the quantitative and qualitative data.

Summarizing, we can make explicit the following research challenges in this field:

1. It is hard to **select the right data** to support the metrics.
2. Some data may be **unavailable**, such as contributor lists.
3. It is hard to **compare ecosystems**, especially when the fundamental data differs.
4. There may be **a lack of historical data**, as data sources frequently do not store data over time.
5. There may **not be a project index**, making it harder to collect all related projects.
6. The ecosystem may be **less transparent**, when it typically is operated in a commercial domain.
7. It is hard to **identify whether a project has been created specifically for an ecosystem** or simply mentions it in the project description.
8. Some **data may be incorrect**, for instance due to faulty data collection on the data collector's side.

9. As there are not yet uniform storage formats for projects, contributors, etc., there are many **data clean-up and preparation challenges**.
10. It is hard to **merge data from different sources**, such as two identities of contributors.
11. It is hard to **cross validate** results from different research methods.

For each future project in ecosystem health we recommend that the researchers specifically address these challenges, to provide insight into their measures and methods.

7. Discussion

The framework is evaluated using the research projects described in the previous sections. There are currently few works on ecosystem health available and the selection of just four research projects is somewhat meager. As these research projects do not fully cover the metrics in the framework, the work cannot be considered completely evaluated. The OSEHO can be further evaluated in the future with more projects that study ecosystem health. The framework, however, is the most complete framework for open source ecosystem health assessment and its contribution lies in the fact that it is the first comprehensive overview of health metrics. Two options are proposed for future evaluation: the use of experts to determine the comprehensiveness of the framework and more case studies illustrating the application of the framework.

The OSEHO is large and comprehensive, but not overly elegant. An interesting question is whether such a framework can be designed to deliver one or a small subset of metrics that provide insight into health. It has been suggested that developer satisfaction is one of the most important metrics in project health. Lakhani and Wolf [18] found the three main motivations of open source developers to be "enjoyment-related intrinsic motivations in the form of a sense of creativity, extrinsic motivations in form of payment, and obligation/community-related intrinsic motivations". Crowston and Scozzi [4] also refer to the personal recognition and possible later employment as an important factor in marshalling competencies. Simultaneously, growth in *lines of code* is highly misleading: a project's quality may improve significantly when a large portion of dead code is removed, when duplicates are replaced, and when a project is significantly refactored. It is considered part of the future work to establish what metrics can be collected to provide a quick insight into ecosystem health, that is also comparable to other ecosystems. Another part of this initiative could be to establish an information source that provides health metrics for as many open source ecosystems as possible, to provide stakeholders with quick insight into the health of the ecosystem they are interested in.

Not all factors in the OSEHO are completely defined: with some of them there is room left for interpretation. Factors such as *market share*, *switching costs to alternatives*, and *artifact quality*, it is up to the researcher to define how these factors are translated to actual metrics. As this work is the first attempt to an OSEHO, this is not a problem, but it will in the future have malignant effects on comparability of health metrics.

This research attempts to abstract from the project level and bring project level metrics to the ecosystem level. This is ambitious: as projects within one ecosystem use different repository management tools, different tools to support developers, and have wildly varying levels of activities, it is hard to find complete data sets and baseline measurements. The four projects, however, overcome these challenges. The platform health project of Lucassen et al. [19], for instance, assumes it to be true that there is missing information, but uses several data sources and claims that completeness of the data is not essential to make predictions about the most successful platform. We hope that future attempts are made at gathering and maintaining complete data sets, to make sure no data is lost for measuring ecosystem health and development.

7.1. Metrics evaluation

The metrics in the OSEHO are of different levels of practicality: some are highly abstract (*organizational maturity*) whereas others are concrete (*number of new projects*). A deliberate degree of freedom is necessary. For example, *organizational maturity* of project organizations can be measured in different ways and there is no consensus yet on how to do so. The OSEHO in this case provides insight into possible metrics, but does not completely define the way in which such a metric could be operationalized. The operationalization of these metrics can be seen as future work and future research challenges.

Several of the metrics open up debate about whether they are beneficial or may be detrimental in some cases. One could argue that forking a project is not a positive development in an ecosystem, as a fork may be a sign of a split in the community. We will argue that forks are a sign of productivity, especially when they occur in large numbers surrounding a project and can be seen as new projects in a lively ecosystem. We do acknowledge, however, that a fork in the main project or platform an ecosystem is based on, may be detrimental to its development.

Another metric up for debate is *a strongly connected core of developers*: in an open source project it may be a sign of a healthy project, but if that core starts to reign without democracy, the project may soon starve due to absence of qualified consenting developers. This indicates that there are cases in which a strongly connected core is not healthy, but in these cases there are other indicators that provide more insight into the situation. If a highly cohesive core, for instance, has little outbound links to other ecosystems and there is little variety in the ecosystem, the ecosystem players should get worried. In most cases, however, a strong core may provide direction and vision in an ecosystem, which is a sign of a robust ecosystem.

van Lingen et al. [27] find that the Wordpress ecosystem has the larger market share, even though Drupal is found to have the most healthy ecosystem. The finding that the ecosystem is healthy even though the market share is not does not invalidate the OSEHO in any way: it is simply a sign that the Wordpress coordinators have chosen a different way to penetrate the market for content management systems, using a highly commercial approach and perhaps also because Wordpress is directed at blogs, whereas Joomla and Drupal are traditionally used for somewhat more complex web applications. Ecosystem health does not say anything about commercial success: a more healthy ecosystem simply indicates that the ecosystem is gaining more developers, activity, and new projects compared to another. What can be said is that a more healthy ecosystem in a set of ecosystems of similar size and in the same domain, is the best choice when choosing to join or invest in an ecosystem. The strategic decisions that are made by an ecosystem coordinator using governance mechanisms (entrance barriers, tenancy prices, possibility of multi-homing, etc.) determine the success of an ecosystem and typically also the success of its coordinating party [1].

In the work of West and Wood [29], the authors discuss the demise of the Symbian ecosystem and defend that although the platform was technologically apt, the business models applied in the ecosystem did not stimulate further growth of the ecosystem. One of the main and most important points made by West and Wood is that the business model of the platform that is central to the ecosystem must be designed as such, that profits and

revenues made from the platform should be re-invested into the platform and surrounding components. One could defend that Symbian was a healthy ecosystem, but was finished off by external factors. As Symbian is a closed-source platform, it is hard to bring these findings to the evaluation of open source platforms, but it is surprising to see that a healthy ecosystem can still be overtaken by a superior ecosystem so quickly. The main recommendation in the context of the OSEHO is to track user loyalty and active developers, as they are the first ones to indicate that another platform or ecosystem is more attractive.

7.2. Conclusion and future work

This paper provides the Open Source Ecosystem Health Operationalization, a framework that is used to establish the health of an open source ecosystem. It is unique because it abstracts from the project level. Its application is explained in detail, illustrated using four research projects from literature, and possible challenges researchers may face are discussed in-depth. The operationalization provides ecosystem researchers with a foundation under their ecosystem health work and they no longer need to start from scratch when establishing the health of an open source ecosystem. To evaluate the framework further, more studies of software ecosystems and their health must be performed. Furthermore, using interviews the framework can further be evaluated by experts.

This article is a call to action for ecosystem health researchers. First, there is a need for historical data that, if not tracked, will get lost over time. Secondly, data quality must constantly be improved, as current data sources are not always accurate. Thirdly, more studies are required in this important field to illustrate how easily data can be gathered and how effectively the data can be used in strategic decision making about an open source ecosystem.

Finally, case studies are an excellent way to further evaluate the OSEHO. There are several case approaches that can be taken. First, it can be attemted to collect all metrics for one particular case. Doing so enables reflection on the framework, for instance in regards to how hard it is to collect certain metrics, analogue to how den Hartigh and his team commented on their ecosystem health operationalization for commercial ecosystems [6]. Secondly, ecosystem participants' observations and perceptions of a developing ecosystem could be compared with the metrics found in the OSEHO. Such a comparison can be used to evaluate which metrics give a realistic image of the health of an ecosystem and which metrics may not be as significant or even contradictory. Thirdly, longtitudinal studies of ecosystems [24], including its metrics, can evaluate the use, effectiveness, and predictive power of the OSEHO.

References

[1] A. Baars, S. Jansen, A framework for software ecosystem governance, in: Software Business, Springer, 2012, pp. 168–180.

[2] O. Barbosa, R. Santos, C. Alves, C. Werner, S. Jansen, A systematic mapping study on software ecosystems through a three-dimensional perspective, in: S. Jansen, M. Cusumano, S. Brinkkemper (Eds.), Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry, Edward Elgar Publishers, 2013.

[3] K. Crowston, J. Howison, H. Annabi, Information systems success in free and open source software development: theory and measures, Software Process: Improvement and Practice 11 (2) (2006) 123–148.

[4] K. Crowston, B. Scozzi, Open source software projects as virtual organisations: competency rallying for software development, IEEE Software 149 (1) (2002) 3–17.

[5] M. Cusumano, Staying Power: Six Enduring Principles for Managing Strategy and Innovation in an Uncertain World (Lessons from Microsoft, Apple, Intel, Google, Toyota and More), Oxford University Press, 2012.

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

[7] D. Dhungana, I. Groher, E. Schludermann, S. Biffl, Guiding principles of natural ecosystems and their applicability to software ecosystems, in: S. Jansen, M. Cusumano, S. Brinkkemper (Eds.), Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry, Edward Elgar Publishers, 2013.

[8] J. Gamalielsson, B. Lundell, B. Lings, The nagios community: an extended quantitative analysis, in: Open Source Software: New Horizons, Springer, 2010, pp. 85–96.

[9] J. Gamalielsson, B. Lundell, B. Lings, Responsiveness as a measure for assessing the health of oss ecosystems, in: Proceedings of the 2nd International Workshop on Building Sustainable Open Source Communities (OSCOMM 2010), 2010, pp. 1–8.

[10] G. Gousios, The ghtorent dataset and tool suite, in: Proceedings of the Tenth International Workshop on Mining Software Repositories, IEEE Press, 2013, pp. 233–236.

[11] N. Haenni, M. Lungu, N. Schwarz, O. Nierstrasz, Categorizing developer information needs in software ecosystems, in: Workshop on Ecosystem Architectures, 2013, pp. 1–5.

[12] R. Hoving, G. Slot, S. Jansen, Python: characteristics identification of a free open source software ecosystem, in: 2013 7th IEEE International Conference on Digital Ecosystems and Technologies (DEST), IEEE, 2013, pp. 13–18.

[13] M. Iansiti, R. Levien, The Keystone Advantage: What the New Dynamics of Business Ecosystems Mean for Strategy, Innovation, and Sustainability, Harvard Business School Press, 2004.

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

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

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

[17] S. Jansen, A. Finkelstein, S. Brinkkemper, A sense of community: a research agenda for software ecosystems, in: 31st International Conference on Software Engineering, New and Emerging Research Track, 2009.

[18] K. Lakhani, R. Wolf, Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects, MIT Press, Cambridge, 2005.

[19] G. Lucassen, K. v. Rooij, S. Jansen, Ecosystem health of cloud paas providers, in: Proceedings of the International Conference on Software Business, Springer, Berlin Heidelberg, 2013.

[20] B. Lundell, B. Forssten, J. Gamalielsson, H. Gustavsson, R. Karlsson, C. Lennerholt, B. Lings, A. Mattsson, E. Olsson, Exploring health within oss ecosystems, in: Proceedings of the First International Workshop on Building Sustainable Open Source Communities, Tampere University of Technology, 2009.

[21] M. Lungu, R. Robbes, M. Lanza, Recovering inter-project dependencies in software ecosystems, in: Proceedings of the IEEE/ACM International Conference on Automated Software Engineering, ACM, 2010, pp. 309–312.

[22] K. Manikas, K.M. Hansen, Software ecosystems–a systematic literature review, J. Syst. Software 85 (2012) 12941306.

[23] K. Manikas, K.M. Hansen, Reviewing the health of software ecosystems: a conceptual framework proposal, in: Proceedings of the International Workshop on Software Ecosystems, 2013.

[24] T. Mens, M. Claes, P. Grosjean, A. Serebrenik, Studying evolving software ecosystems based on ecological models, in: Evolving Software Systems, Springer, Berlin Heidelberg, 2014, pp. 297–326.

[25] T. Mens, M. Goeminne, Analysing ecosystems for open source software developer communities, in: S. Jansen, M. Cusumano, S. Brinkkemper (Eds.), Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry, Edward Elgar Publishers, 2013.

[26] S. Syed, S. Jansen, On clusters in open source ecosystems, in: Proceedings of the International Workshop on Software Ecosystems, 2013.

[27] S. van Lingen, A. Palomba, G. Lucassen, On the software ecosystem health of open source content management systems, in: the Proceedings of the 5th Workshop on Software Ecosystems, 2013.

[28] J. Waltl, J. Henkel, C.Y. Baldwin, Ip modularity in software ecosystems: how sugarcrms ip and business model shape its product architecture, in: Software Business, Springer, 2012, pp. 94–106.

[29] J. West, D. Wood, Evolving an open ecosystem: the rise and fall of the symbian platform, Advances in Strategic Management 30 (2013) 27–67.

[30] A. Wiggins, J. Howison, K. Crowston, Heartbeat: measuring active user base and potential user interest in floss projects, in: Open Source Ecosystems: Diverse Communities Interacting, 2009, 94–104.

[31] D. Wynn, Assessing the health of an open source ecosystem, in: Emerging Free and Open Source Software Practices, Idea Group Publishing, 2007.